WebSphere MQ for i5/OS

System Administration Guide

Version 7.0

SC34-6930-01

WebSphere MQ for i5/OS

System Administration Guide

Version 7.0

SC34-6930-01

Note
Before using this information and the product it supports, be sure to read the general information under notices at the back of this book.

Second edition (January 2009)

This edition of the book applies to the following:

- IBM WebSphere MQ for i5/OS, Version 7.0

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

Contents

Figures vii

Tables ix

Chapter 1. Introduction to WebSphere MQ 1
WebSphere MQ and message queuing 1
Time-independent applications 1
Message-driven processing 1
Messages and queues 2
What is a message? 2
What is a queue? 3
Objects 3
Object names 4
Managing objects 4
Object attributes 5
WebSphere MQ queue managers 5
WebSphere MQ queues 6
Authentication information objects 9
Channels 9
Client connection channels. 9
Clusters 9
Listeners 10
Namelists 10
Process definitions 10
Services. 11
System default objects 11
Clients and servers 11
WebSphere MQ applications in a client-server environment 11
Extending queue manager facilities 12
User exits 12
Security 12
Transactional support 12
Daylight saving time 13

Chapter 2. Managing WebSphere MQ for i5/OS using CL commands 15
WebSphere MQ applications 15
WebSphere MQ for i5/OS CL commands 15
General usage tips 18
Before you start 19
Starting a local queue manager 19
Creating WebSphere MQ objects 20
Examples of creating a local queue 20
Examples of creating a remote queue 22
Creating a transmission queue 23
Creating an initiation queue 24
Creating an alias queue 24
Creating a model queue 24
Altering queue manager attributes. 24
Working with local queues 24
Defining a local queue. 25
Defining a dead-letter queue 25
Displaying default object attributes 26
Copying a local queue definition 26
Changing local queue attributes 26
Clearing a local queue 27
Deleting a local queue 27
Enabling large queues 27
Working with alias queues 27
Defining an alias queue 28
Using other commands with alias queues . . . 28
Working with model queues. 29
Defining a model queue 29
Using other commands with model queues. . . 29
Working with triggering 29
What is triggering? 30
What is the trigger monitor?. 30
Altering the job submission attributes of the trigger monitor 30
Setting up objects for triggering 31
Communicating between two systems 33

Chapter 3. Alternative ways of administering WebSphere MQ 35
Local and remote administration 35
Administration using MQSC commands 36
MQSC command files 37
Administration using PCF commands 37
Attributes in MQSC and PCF commands . . . 38
Escape PCFs 38
Using the MQAI to simplify the use of PCFs . . 38
Using the WebSphere MQ Explorer with WebSphere MQ for i5/OS 39
What you can do with the WebSphere MQ Explorer 40
Required definitions for administration 40
Managing the command server for remote administration 40
Starting the command server 41
Displaying the status of the command server . . 41
Stopping a command server 41
Instrumentation events 42

Chapter 4. Work management 43
Description of WebSphere MQ tasks 43
WebSphere MQ work management objects 44
How WebSphere MQ uses the work management objects 45
The WebSphere MQ message queue 47
Configuring work management. 48

Chapter 5. Protecting WebSphere MQ objects. 51
Security considerations 51
Understanding the Object Authority Manager . . . 52
Resources you can protect with the OAM . . . 52
WebSphere MQ authorities 52

Access authorities for WebSphere MQ objects . . 53
Understanding the authorization specification tables 59
MQI authorizations 60
Administration authorizations 63
Authorizations for MQSC commands in escape PCFs 63
Generic OAM profiles 74
Using wildcard characters 74
Profile priorities 75
Specifying the installed authorization service . . . 75
Working without authority profiles 75
Working with authority profiles 76
WRKMQMAUT 76
WRKMQMAUTD 77
Object Authority Manager guidelines 79
Queue manager directories 79
Queues 79
Alternate-user authority 79
Context authority 80
Remote security considerations 80
Channel command security 81

Chapter 6. The WebSphere MQ dead-letter queue handler 83
Invoking the DLQ handler 83
The DLQ handler rules table 84
Control data 84
Rules (patterns and actions) 85
Rules table conventions 88
Processing the rules table 90
Ensuring that all DLQ messages are processed . 90
An example DLQ handler rules table 91

Chapter 7. Backup, recovery, and restart 93
WebSphere MQ for i5/OS journals 93
WebSphere MQ for i5/OS journal usage 95
Media images 96
Recovery from media images 97
Checkpoints 97
Backups of WebSphere MQ for i5/OS data 98
Journal management 99
Restoring a complete queue manager (data and journals) 102
Restoring journal receivers for a particular queue manager 102
Performance and failover considerations 103
Using SAVLIB to save WebSphere MQ libraries . . 104

Chapter 8. Analyzing problems 105
Preliminary checks 105
Problem characteristics 107
Can you reproduce the problem? 107
Is the problem intermittent? 108
Problems with commands 108
Does the problem affect all users of the WebSphere MQ for i5/OS application? 108
Does the problem affect specific parts of the network? 108
Does the problem occur only on WebSphere MQ 109
Does the problem occur at specific times of the day? 109
Have you failed to receive a response from a command? 109
Determining problems with WebSphere MQ applications 110
Are some of your queues working? 110
Does the problem affect only remote queues? 111
Does the problem affect messages? 111
Unexpected messages are received when using distributed queues 113
Obtaining diagnostic information 113
Using WebSphere MQ for i5/OS trace 114
Formatting trace output 117
Error logs 117
Log files 117
Early errors 118
Operator messages 118
An example WebSphere MQ error log 119
Dead-letter queues 120
First Failure Support Technology (FFST) 120
Performance considerations. 122
Application design considerations 122
Number of threads in use 123
Specific performance problems 123

Chapter 9. Configuring WebSphere MQ 125
WebSphere MQ configuration files 125
Editing configuration files 125
The WebSphere MQ configuration file mqs.ini 126
Queue manager configuration files qm.ini . . . 126
Attributes for changing WebSphere MQ configuration information 127
The AllQueueManagers stanza 127
The DefaultQueueManager stanza 128
The ExitProperties stanza 128
The QueueManager stanza 129
Changing queue manager configuration information 130
The Log stanza 130
The Channels stanza 130
The queue manager error log stanza. 132
The TCP stanza. 133
API exits 134
Why use API exits. 135
How you use API exits 135
What happens when an API exit runs? 136
Configuring API exits 136
Example mqs.ini and qm.ini files 138

Chapter 10. Installable services and components 141
Why installable services? 141
Functions and components 142
Entry-points 142
Return codes 143
Component data 143
Initialization 143
Primary initialization 143

Secondary initialization 143
Primary termination 144
Secondary termination 144
Configuring services and components 144
Service stanza format. 144
Service component stanza format. 144
Creating your own service component 145
Authorization service. 145
Object authority manager (OAM). 146
Configuring authorization service stanzas . . . 146
Authorization service interface 147
Installable services interface reference information 148
How the functions are shown 148
MQZEP – Add component entry point 149
MQHCONFIG – Configuration handle 150
PMQFUNC – Pointer to function 150
MQZ_AUTHENTICATE_USER – Authenticate user 150
MQZ_CHECK_AUTHORITY – Check authority 153
MQZ_COPY_ALL_AUTHORITY – Copy all authority 157
MQZ_DELETE_AUTHORITY – Delete authority 160
MQZ_ENUMERATE_AUTHORITY_DATA – Enumerate authority data 162
MQZ_FREE_USER – Free user. 165
MQZ_GET_AUTHORITY – Get authority . . . 167
MQZ_GET_EXPLICIT_AUTHORITY – Get explicit authority 170
MQZ_INIT_AUTHORITY – Initialize authorization service 173
MQZ_INQUIRE – Inquire authorization service 175
MQZ_REFRESH_CACHE – Refresh all authorizations 179
MQZ_SET_AUTHORITY – Set authority . . . 181
MQZ_TERM_AUTHORITY – Terminate authorization service 184
MQZAC – Application context 185
MQZAD – Authority data 188
MQZED – Entity descriptor 191
MQZFP – Free parameters 192
MQZIC – Identity context 193

Chapter 11. The CL commands 197
Connect MQ (CCTMQM) 197
Change Message Queue Manager (CHGMQM) . . 197
Change MQ AuthInfo object (CHGMQMAUTI) . . 220
Change MQ Channel (CHGMQMCHL). 222
Change MQ Listener (CHGMQMLSR) 252
Change MQ Namelist (CHGMQMNL) 255
Change MQ Process (CHGMQMPRC) 256
Change MQ Queue (CHGMQMQ) 260
Change MQ Subscription (CHGMQMSUB) . . . 282
Change MQ Service (CHGMQMSVC) 289
Change MQ Topic (CHGMQMTOP) 294
Clear MQ Pub/Sub Broker (CLRMQMBRK) . . . 300
Clear MQ Queue (CLRMQMQ) 302
Clear MQ Topic String (CLRMQMTOP) 303
Copy MQ AuthInfo object (CPYMQMAUTI) . . . 304
Copy MQ Channel (CPYMQMCHL). 308
Copy MQ Listener (CPYMQMLSR) 338
Copy MQ Namelist (CPYMQMNL) 342
Copy MQ Process (CPYMQMPRC) 344
Copy MQ Queue (CPYMQMQ) 348
Copy MQ Subscription (CPYMQMSUB) 370
Copy MQ Service (CPYMQMSVC) 378
Copy MQ Topic (CPYMQMTOP) 383
Create Message Queue Manager (CRTMQM) . . . 390
Create MQ AuthInfo object (CRTMQMAUTI). . . 395
Create MQ Channel (CRTMQMCHL) 398
Create MQ Listener (CRTMQMLSR). 429
Create MQ Namelist (CRTMQMNL). 432
Create MQ Process (CRTMQMPRC) 434
Create MQ Queue (CRTMQMQ) 439
Create MQ Subscription (CRTMQMSUB) 461
Create MQ Service (CRTMQMSVC) 469
Create MQ Topic (CRTMQMTOP) 473
Convert MQ Data Type (CVTMQMDTA) 480
Delete Message Queue Manager (DLTMQM) . . . 483
Delete MQ AuthInfo object (DLTMQMAUTI). . . 483
Delete MQ Pub/Sub Broker (DLTMQMBRK) . . . 485
Delete MQ Channel (DLTMQMCHL) 485
Delete MQ Listener (DLTMQMLSR). 487
Delete MQ Namelist (DLTMQMNL). 488
Delete MQ Process (DLTMQMPRC) 489
Delete MQ Queue (DLTMQMQ) 491
Delete MQ Subscription (DLTMQMSUB) 492
Delete MQ Service (DLTMQMSVC) 493
Delete MQ Topic (DLTMQMTOP) 494
Disconnect MQ (DSCMQM) 496
Display Message Queue Manager (DSPMQM) . . 496
Display MQ Object Authority (DSPMQMAUT) . . 497
Display MQ AuthInfo object (DSPMQMAUTI) . . 500
Display MQ Pub/Sub Broker (DSPMQMBRK) . . 501
Display MQ Channel (DSPMQMCHL) 502
Display MQ Command Server (DSPMQMCSVR) 504
Display MQ Listener (DSPMQMLSR) 505
Display MQ Namelist (DSPMQMNL) 507
Display MQ Object Names (DSPMQMOBJN) . . . 508
Display MQ Process (DSPMQMPRC) 511
Display MQ Queue (DSPMQMQ) 512
Display MQ Route Information (DSPMQMRTE) 514
Display Queue Manager Status (DSPMQMSTS) . . 522
Display MQ Subscription (DSPMQMSUB). . . . 523
Display MQ Service (DSPMQMSVC) 525
Display MQ Topic (DSPMQMTOP) 526
Display MQ Version (DSPMQMVER) 527
End Message Queue Manager (ENDMQM) . . . 528
End MQ Pub/Sub Broker (ENDMQMBRK) . . . 531
End MQ Channel (ENDMQMCHL) 532
End Queue Manager Connection (ENDMQMCONN) 534
End MQ Command Server (ENDMQMCSVR) . . 536
End MQ Listeners (ENDMQMLSR) 537
End MQ Service (ENDMQMSVC) 538
Grant MQ Object Authority (GRTMQMAUT). . . 540
Ping MQ Channel (PNGMQMCHL) 546
Record MQ Object Image (RCDMQMIMG) . . . 547
Recreate MQ Object (RCRMQMOBJ) 550
Refresh WebSphere MQ Authority (RFRMQMAUT) 553
Refresh MQ Cluster (RFRMQMCL) 554
Resume Cluster Queue Manager (RSMMQMCLQM) 555

Reset MQ Channel (RSTMQMCHL) 557
Reset Cluster (RSTMQMCL) 558
Resolve MQ Channel (RSVMQMCHL) 560
RUNMQSC (RUNMQSC) 562
Revoke MQ Object Authority (RVKMQMAUT) . . 563
Suspend Cluster Queue Manager (SPDMQMCLQM) 569
Start Message Queue Manager (STRMQM) . . . 570
Start MQ Pub/Sub Broker (STRMQMBRK) . . . 573
Start MQ Channel (STRMQMCHL) 574
Start MQ Channel Initiator (STRMQMCHLI) . . . 575
Start MQ Command Server (STRMQMCSVR) . . 576
Start WebSphere MQ DLQ Handler (STRMQMDLQ) 577
Start MQ Listener (STRMQMLSR) 580
Start WebSphere MQ Commands (STRMQMMQSC) 582
Start MQ Service (STRMQMSVC). 584
Start MQ Trigger Monitor (STRMQMTRM) . . . 585
Trace MQ (TRCMQM) 586
Work with MQ Queue Manager (WRKMQM) . . 593
Work with MQ Authority (WRKMQMAUT) . . . 595
Work with MQ Authority Data (WRKMQMAUTD) 597
Work with AuthInfo objects (WRKMQMAUTI) . . 600
Work with MQ Channels (WRKMQMCHL) . . . 603
Work with MQ Channel Status (WRKMQMCHST) 613
Work with MQ Clusters (WRKMQMCL) 620
Work with MQ Cluster Queues (WRKMQMCLQ) 629
Work with MQ Connections (WRKMQMCONN) 634
Work with MQ Listeners (WRKMQMLSR). . . . 638
Work with MQ Messages (WRKMQMMSG) . . . 642
Work with MQ Namelist (WRKMQMNL) 644
Work with MQ Processes (WRKMQMPRC) . . . 646
Work with MQ Queues (WRKMQMQ) 650
Work with Queue Status (WRKMQMQSTS) . . . 662
Work with MQ Subscriptions (WRKMQMSUB) . . 665
Work with MQ Service object (WRKMQMSVC) . . 670
Work with MQ Topics (WRKMQMTOP) 674
Work with MQ Transactions (WRKMQMTRN) . . 679

Chapter 12. WebSphere MQ names and default objects 681
WebSphere MQ object names 681
Understanding WebSphere MQ queue manager library names 681
Understanding WebSphere MQ IFS directories and files 682
IFS queue manager name transformation . . . 682
Object name transformation 682
System and default objects 683

Chapter 13. Sample resource definitions. 687

Chapter 14. Quiescing WebSphere MQ for i5/OS 691
Quiescing WebSphere MQ for i5/OS 691
ENDMQM parameter ENDCCTJOB(*YES). . . 691
Shutting down a single queue manager . . . 691
Shutting down all queue managers 693

Notices 695

Index 699

Sending your comments to IBM . . . 705

Figures

1. Create MQM Queue initial panel 21
2. Work with MQM Queues panel 21
3. Display MQ Process panel 31
4. Work with queue managers results panel 36
5. Extract from the MQSC command file, myprog.in 37
6. Display MQM Command Server panel 41
7. Work with MQM Authority panel – input display 76
8. Work with MQM Authority panel – results display 77
9. Work with MQM Authority Data input panel 78
10. Work with MQM Authority Data output panel 78
11. Sequence of events when updating MQM objects 96
12. WebSphere MQ for i5/OS journaling 100
13. Extract from a WebSphere MQ error log 119
14. Example of a WebSphere MQ configuration file 139
15. Example queue manager configuration file 140
16. WebSphere MQ for i5/OS authorization service stanzas in qm.ini 147

Tables

1. WebSphere MQ tasks. 44
2. Work management objects 45
3. Work management objects created for a queue manager 45
4. Authorizations for MQI calls 57
5. Authorizations for context calls 58
6. Authorizations for MQSC and PCF calls 58
7. Authorizations for generic operations 58
8. Security authorization needed for MQCONN calls 60
9. Security authorization needed for MQOPEN calls 61
10. Security authorization needed for MQPUT1 calls 61
11. Security authorization needed for MQCLOSE calls 62
12. List of possible ISO CCSIDs. 128
13. Authorization service components summary 141
14. Queue manager library names 681
15. System and default objects: queues 683
16. System and default objects: channels 684
17. System and default objects: authentication information objects. 684
18. System and default objects: listeners 684
19. System and default objects: namelists 685
20. System and default objects: processes 685
21. System and default objects: services 685

Chapter 1. Introduction to WebSphere MQ

This section introduces the WebSphere® MQ for i5/OS® Version 7.0 product from an administrator's perspective, and describes the basic concepts of WebSphere MQ and messaging.

It contains these topics:

- "WebSphere MQ and message queuing"
- "Messages and queues" on page 2
- "Objects" on page 3
- "System default objects" on page 11
- "Clients and servers" on page 11
- "Extending queue manager facilities" on page 12
- "Security" on page 12
- "Transactional support" on page 12

WebSphere MQ and message queuing

WebSphere MQ allows application programs to use **message queuing** to participate in message-driven processing. Application programs can communicate across different platforms by using the appropriate message queuing software products. For example, i5/OS and z/OS® applications can communicate through WebSphere MQ for i5/OS and WebSphere MQ for z/OS respectively. The applications are shielded from the mechanics of the underlying communications.

WebSphere MQ products implement a common application programming interface known as the **message queue interface** (or MQI) whatever platform the applications are run on. This makes it easier for you to port application programs from one platform to another.

The MQI is described in detail in the WebSphere MQ Application Programming Reference.

Time-independent applications

With message queuing, the exchange of messages between the sending and receiving programs is independent of time. This means that the sending and receiving application programs are decoupled; the sender can continue processing without having to wait for the receiver to acknowledge receipt of the message. In fact, the target application does not even have to be running when the message is sent. It can retrieve the message after it is has been started.

Message-driven processing

Upon arrival on a queue, messages can automatically start an application using a mechanism known as **triggering**. If necessary, the applications can be stopped when the message (or messages) have been processed.

Messages and queues

Messages and queues are the basic components of a message queuing system.

What is a message?

A **message** is a string of bytes that is meaningful to the applications that use it. Messages are used for transferring information from one application program to another (or to different parts of the same application). The applications can be running on the same platform, or on different platforms.

WebSphere MQ messages have two parts:

- **The application data**

  The content and structure of the application data is defined by the application programs that use them.
- **A message descriptor**

  The message descriptor identifies the message and contains additional control information such as the type of message, and the priority assigned to the message by the sending application.

  The format of the message descriptor is defined by WebSphere MQ. For a complete description of the message descriptor, see the WebSphere MQ Application Programming Reference.

Message lengths

The default maximum message length is 4 MB, although you can increase this to a maximum length of 100 MB (where 1 MB equals 1 048 576 bytes). In practice, the message length is limited by:

- The maximum message length defined for the receiving queue
- The maximum message length defined for the queue manager
- The maximum message length defined by the queue
- The maximum message length defined by either the sending or receiving application
- The amount of storage available for the message

It can take several messages to send all the information that an application requires.

How do applications send and receive messages?

Application programs send and receive messages using **MQI calls**.

For example, to put a message onto a queue, an application:

1. Connects to the queue manager by issuing an MQI MQCONN call
2. Opens the required queue by issuing an MQI MQOPEN call
3. Issues an MQI MQPUT call to put the message onto the queue

   Note: Another application can retrieve the message from the same queue by issuing an MQI MQGET call.
4. Closes the specified queue by issuing an MQI MQCLOSE call
5. Disconnects from the queue manager by issuing an MQI MQDISC call

For more information about MQI calls, see the WebSphere MQ Application Programming Reference.

What is a queue?

A **queue** is a data structure used to store messages. The messages are put on the queue by application programs, or by a **queue manager** as part of its normal operation.

Each queue is owned by a queue manager. The queue manager maintains the queues it owns and stores all the messages it receives onto the appropriate queues.

WebSphere MQ supports queues over 2 GB in size; "Enabling large queues" on page 27 discusses this in more detail. For information about planning the amount of storage you need for queues, visit the WebSphere MQ Web site for platform-specific performance reports: http://www.ibm.com/software/integration/wmq/

Predefined queues and dynamic queues

Queues can be characterized by the way that they are created:

- **Predefined queues** are created by an administrator using the appropriate WebSphere MQ script (MQSC) commands. Predefined queues are permanent; they exist independently of the applications that use them and survive WebSphere MQ restarts.
- **Dynamic queues** are created when an application issues an OPEN request specifying the name of a **model queue**. The queue created is based on a *template queue definition*, which is the model queue. You can create a model queue using the MQSC DEFINE QMODEL command. The attributes of a model queue, for example the maximum number of messages that can be stored on it, are inherited by any dynamic queue that is created from it.

  Model queues have an attribute that specifies whether the dynamic queue is to be permanent or temporary. Permanent queues survive application and queue manager restarts; temporary queues are lost on restart.

For further information about MQSC, see WebSphere MQ Script (MQSC) Command Reference.

Retrieving messages from queues

Suitably authorized applications can retrieve messages from a queue according to the following retrieval algorithms:

- First-in-first-out (FIFO)
- Message priority, as defined in the message descriptor. Messages that have the same priority are retrieved on a FIFO basis.
- A program request for a specific message.

The MQGET request from the application determines the method used.

Objects

Many of the tasks described in this book involve manipulating WebSphere MQ **objects**.

In WebSphere MQ, the object types include queue managers, queues, process definitions, namelists, channels, client connection channels, listeners, services, and authentication information objects.

The manipulation or *administration* of objects includes:

- Starting and stopping queue managers.
- Creating objects, particularly queues, for applications.
- Working with channels to create communication paths to queue managers on other (remote) systems. This is described in detail in WebSphere MQ Intercommunication.
- Creating *clusters* of queue managers to simplify the overall administration process, or to achieve workload balancing. This is described in detail in WebSphere MQ Queue Manager Clusters.

This book contains detailed information about administration in the following chapters:

- Chapter 2, "Managing WebSphere MQ for i5/OS using CL commands," on page 15
- Chapter 3, "Alternative ways of administering WebSphere MQ," on page 35

You can also administer WebSphere MQ for i5/OS from a Windows® or Linux® (x86 platform) machine using the WebSphere MQ Explorer.

Object names

The naming convention adopted for WebSphere MQ objects depends on the object.

Each instance of a queue manager is known by its name. This name must be unique within the network of interconnected queue managers, so that one queue manager can unambiguously identify the target queue manager to which any given message is sent.

For the other types of object, each object has a name associated with it and can be referenced by that name. These names must be unique within one queue manager and object type. For example, you can have a queue and a process with the same name, but you cannot have two queues with the same name.

In WebSphere MQ, names can have a maximum of 48 characters, with the exception of *channels* and *client connection channels* that have a maximum of 20 characters. For more information about names, see "WebSphere MQ object names" on page 681.

Managing objects

You can manage objects using the native i5/OS menus.

You can create, alter, display, and delete objects using:

- WebSphere MQ for i5/OS CL commands
- WebSphere MQ script (MQSC) commands, which can be typed in from a keyboard or read from a file
- Programmable command format (PCF) messages, which can be used in an automation program
- WebSphere MQ Administration Interface (MQAI) calls in a program

For more information about these methods, see Chapter 3, "Alternative ways of administering WebSphere MQ," on page 35.

You can also administer WebSphere MQ for i5/OS from a Windows machine using the WebSphere MQ Explorer (see "Using the WebSphere MQ Explorer with WebSphere MQ for i5/OS" on page 39).

Object attributes

The properties of an object are defined by its attributes. Some you can specify, others you can only view. For example, the maximum message length that a queue can accommodate is defined by its *MaxMsgLength* attribute; you can specify this attribute when you create a queue. The *DefinitionType* attribute specifies how the queue was created; you can only display this attribute.

In WebSphere MQ, there are three ways of referring to an attribute:

- Using its CL parameter name, for example, MAXMSGLEN
- Using its PCF name, for example, *MaxMsgLength*.
- Using its MQSC name, for example, MAXMSGL.

The formal name of an attribute is its PCF name. Because using the CL interface is an important part of this book, you are more likely to see the CL name in examples than the PCF name of a given attribute.

WebSphere MQ queue managers

A *queue manager* provides queuing services to applications, and manages the queues that belong to it. It ensures that:

- Object attributes are changed according to the commands received.
- Special events such as trigger events or instrumentation events are generated when the appropriate conditions are met.
- Messages are put on the correct queue, as requested by the application making the MQPUT call. The application is informed if this cannot be done, and an appropriate reason code is given.

Each queue belongs to a single queue manager and is said to be a *local queue* to that queue manager.

The queue manager to which an application is connected is said to be the local queue manager for that application. For the application, the queues that belong to its local queue manager are local queues.

A *remote queue* is a queue that belongs to another queue manager.

A *remote queue manager* is any queue manager other than the local queue manager. A remote queue manager exists on a remote machine across the network, or on the same machine as the local queue manager.

WebSphere MQ for i5/OS supports multiple queue managers on the same machine.

A queue manager object can be used in some MQI calls. For example, you can inquire about the attributes of the queue manager object using the MQI call MQINQ.

Note: You cannot put messages on a queue manager object; messages are always put on queue objects, not on queue manager objects.

WebSphere MQ queues

Queues are defined to WebSphere MQ using:

- The native i5/OS CRTMQMQ CL command
- The appropriate MQSC DEFINE command
- The PCF Create Queue command

Note: The WebSphere MQ process definition, channel, and namelist objects can be defined in a similar manner.

The commands specify the type of queue and its attributes. For example, a local queue object has attributes that specify what happens when applications reference that queue in MQI calls. Examples of attributes are:

- Whether applications can retrieve messages from the queue (GET enabled).
- Whether applications can put messages on the queue (PUT enabled).
- Whether access to the queue is exclusive to one application or shared between applications.
- The maximum number of messages that can be stored on the queue at the same time (maximum queue depth).
- The maximum length of messages that can be put on the queue.

For further details about defining queue objects, see the WebSphere MQ Script (MQSC) Command Reference or WebSphere MQ Programmable Command Formats and Administration Interface.

Using queue objects

There are four types of queue object available in WebSphere MQ. Each type of object can be manipulated by the product commands and is associated with real queues in different ways.

1. **Local queue object** A local queue object identifies a local queue belonging to the queue manager to which the application is connected. All queues are local queues in the sense that each queue belongs to a queue manager and, for that queue manager, the queue is a local queue.
2. **A remote queue object**

   A remote queue object identifies a queue belonging to another queue manager. This queue must be defined as a local queue to that queue manager. The information you specify when you define a remote queue object allows the local queue manager to find the remote queue manager, so that any messages destined for the remote queue go to the correct queue manager.

   Before applications can send messages to a queue on another queue manager, you must have defined a transmission queue and channels between the queue managers, **unless** you have grouped one or more queue managers together into a *cluster*. For more information about clusters, see WebSphere MQ Queue Manager Clusters.
3. **An alias queue object**

   An alias queue allows applications to access a queue by referring to it indirectly in MQI calls. When an alias queue name is used in an MQI call, the name is resolved to the name of either a local or a remote queue at run time.

This allows you to change the queues that applications use without changing the application in any way. You just change the alias queue definition to reflect the name of the new queue to which the alias resolves.

An alias queue is not a queue, but an object that you can use to access another queue.

4. **A model queue object**

   A model queue defines a set of queue attributes that are used as a template for creating a dynamic queue. Dynamic queues are created by the queue manager when an application issues an MQOPEN request specifying a queue name that is the name of a model queue. The dynamic queue that is created in this way is a local queue whose attributes are taken from the model queue definition. The dynamic queue name can be specified by the application or the queue manager can generate the name and return it to the application.

   Dynamic queues defined in this way are either temporary queues, which do not survive product restarts, or permanent queues, which do.

Specific local queue types and their uses

WebSphere MQ uses some local queues for specific purposes related to its operation. These are:

- **Application queues**

  This is a queue that is used by an application through the MQI. It can be a local queue on the queue manager to which an application is linked, or it can be a remote queue that is owned by another queue manager.

  Applications can put messages on local or remote queues. However, they can only get messages from a local queue.

- **Initiation queues**

  Initiation queues are queues that are used in triggering. A queue manager puts a trigger message on an initiation queue when a trigger event occurs. A trigger event is a logical combination of conditions that is detected by a queue manager. For example, a trigger event might be generated when the number of messages on a queue reaches a predefined depth. This event causes the queue manager to put a trigger message on a specified initiation queue. This trigger message is retrieved by a *trigger monitor*, a special application that monitors an initiation queue. The trigger monitor then starts up the application program that was specified in the trigger message.

  If a queue manager is to use triggering, at least one initiation queue must be defined for that queue manager.

  See “Working with triggering” on page 29 For more information about triggering, see the WebSphere MQ Application Programming Guide.

- **Transmission queues**

  Transmission queues are queues that temporarily stores messages that are destined for a remote queue manager. You must define at least one transmission queue for each remote queue manager to which the local queue manager is to send messages directly. These queues are also used in remote administration. For information about the use of transmission queues in distributed queuing, see WebSphere MQ Intercommunication.

- **Cluster transmission queues**

  Each queue manager within a cluster has a cluster transmission queue called SYSTEM.CLUSTER.TRANSMIT.QUEUE. A definition of this queue is created by default on every queue manager on WebSphere MQ for AIX®, i5/OS, HP-UX, Solaris, and Windows.

A queue manager that is part of the cluster can send messages on the cluster transmission queue to any other queue manager that is in the same cluster.

Cluster queue managers can communicate with queue managers that are not part of the cluster. To do this, the queue manager must define channels and a transmission queue to the other queue manager in the same way as in a traditional distributed-queuing environment.

For more information on using clusters, see WebSphere MQ Queue Manager Clusters.

- **Dead-letter queues**

  A dead-letter queue is a queue that stores messages that cannot be routed to their correct destinations. This occurs when, for example, the destination queue is full. The supplied dead-letter queue is called SYSTEM.DEAD.LETTER.QUEUE. These queues are sometimes referred to as undelivered-message queues.

  A dead-letter queue is defined by default when each queue manager is created. However, you **must** ensure that the queue manager on which this queue resides points to the dead-letter queue that it is going to use.

  The following command creates an undelivered-message queue on queue manager neptune.queue.manager:

  ```
  CRTMQM MQMNAME(neptune.queue.manager) UDLMSGQ(ANOTHERDLQ)
  ```

- **Command queues**

  The command queue, named SYSTEM.ADMIN.COMMAND.QUEUE, is a local queue to which suitably authorized applications can send WebSphere MQ commands for processing. These commands are then retrieved by a WebSphere MQ component called the command server. The command server validates the commands, passes the valid ones on for processing by the queue manager, and returns any responses to the appropriate reply-to queue.

  A command queue is created automatically for each queue manager when that queue manager is created.

- **Reply-to queues**

  When an application sends a request message, the application that receives the message can send back a reply message to the sending application. This message is put on a queue, called a reply-to queue, which is normally a local queue to the sending application. The name of the reply-to queue is specified by the sending application as part of the message descriptor.

- **Event queues**

  WebSphere MQ for i5/OS supports instrumentation events, which can be used to monitor queue managers independently of MQI applications. Instrumentation events can be generated in several ways, for example:

  – An application attempting to put a message on a queue that is not available or does not exist.
  – A queue becoming full.
  – A channel being started.

  When an instrumentation event occurs, the queue manager puts an event message on an event queue. This message can then be read by a monitoring application, which informs an administrator or initiate some remedial action if the event indicates a problem.

  Note: Trigger events are quite different from instrumentation events in that trigger events are not caused by the same conditions, and do not generate event messages.

For more information about instrumentation events, see Monitoring WebSphere MQ.

Authentication information objects

The queue manager *authentication information object* forms part of WebSphere MQ support for Secure Sockets Layer (SSL) security. It provides the definitions needed to check certificate revocation lists (CRLs) using LDAP servers. CRLs allow Certification Authorities to revoke certificates that can no longer be trusted.

This book introduces the **WRKMQMAUTI**, **DSPMQMAUTI**, **CRTMQMAUTI**, **CPYMQMAUTI**, **CHGMQMAUTI**, and **DLTMQMAUTI** commands for use with the authentication information object. For an overview of SSL and the use of authentication information objects, see WebSphere MQ Security.

Channels

Channels are objects that provide a communication path from one queue manager to another. Channels are used in distributed message queuing to move messages from one queue manager to another. They shield applications from the underlying communications protocols. The queue managers exist on the same, or different, platforms. For queue managers to communicate with one another, you must define one channel object at the queue manager that is to send messages, and another, complementary one, at the queue manager that is to receive them.

Use the WebSphere MQ for i5/OS CRTMQMCHL CL command, the MQSC command DEFINE CHANNEL, or the PCF command Create Channel to create a channel definition.

Note: Clustering automates some of these tasks for you.

For information on channels and how to use them, see WebSphere MQ Intercommunication.

Client connection channels

Client connection channels are objects that provide a communication path from a WebSphere MQ client to a queue manager.

Client connection channels are used in distributed queuing to move messages between a queue manager and a client. They shield applications from the underlying communications protocols. The client might exist on the same, or different, platform to the queue manager.

For information on client connection channels and how to use them, see WebSphere MQ Intercommunication and WebSphere MQ Clients.

Clusters

In a traditional WebSphere MQ network using distributed queuing, every queue manager is independent. If one queue manager needs to send messages to another queue manager it must have defined a transmission queue, a channel to the remote queue manager, and a remote queue definition for every queue to which it wants to send messages.

A *cluster* is a group of queue managers set up in such a way that the queue managers can communicate directly with one another over a single network, without the need for complex transmission queue, channel, and queue definitions.

For information about clusters, see WebSphere MQ Queue Manager Clusters.

Listeners

Listener objects are used to accept incoming network requests from remote queue managers, or client applications. Once accepted, a listener starts the associated receiver or server connection channels to allow the queue manager to receive messages.

An alternative to listener objects, are listener processes. Listener processes are started using the **STRMQMLSR** control command. If you start a listener as a process, you do not inherit the benefits of listener objects, such as:

- You cannot guarantee the running environment of the listener.
- You cannot automatically startup, and shutdown, the listener with the queue manager.

Namelists

A namelist is a WebSphere MQ object that contains a list of other WebSphere MQ objects. Typically, namelists are used by applications such as trigger monitors, where they are used to identify a group of queues. The advantage of using a namelist is that it is maintained independently of applications; that is, it can be updated without stopping any of the applications that use it. Also, if one application fails, the namelist is not affected and other applications can continue using it.

Namelists are also used with queue manager clusters so that you can maintain a list of clusters referenced by more than one WebSphere MQ object.

Use the WebSphere MQ for i5/OS CRTMQMNL CL command, the MQSC command DEFINE NAMELIST, or the PCF command Create Namelist to create a namelist definition.

Process definitions

A *process definition object* defines an application that is to be started in response to a trigger event on a WebSphere MQ queue manager. See the "Initiation queues" entry under "Specific local queue types and their uses" on page 7 for more information.

The process definition attributes include the application ID, the application type, and data specific to the application.

Use the WebSphere MQ for i5/OS CRTMQMPRC CL command, the MQSC command DEFINE PROCESS, or the PCF command Create Process to create a process definition.

Services

Service objects contain the definitions of one or more commands to be run when a queue manager is started, or stopped. By defining service objects an administrator can automate the starting and stopping of tasks, or certain objects associated with the queue manager.

Use the WebSphere MQ for i5/OS CRTMQMSVC CL command, the MQSC command DEFINE SERVICE, or the PCF command Create Service to create a service definition.

System default objects

The *system default objects* are a set of object definitions that are created automatically whenever a queue manager is created. You can copy and modify any of these object definitions for use in applications at your installation.

Default object names have the stem SYSTEM.DEF; for example, the default local queue is SYSTEM.DEFAULT.LOCAL.QUEUE, and the default receiver channel is SYSTEM.DEF.RECEIVER. You cannot rename these objects; default objects of these names are required.

When you define an object, any attributes that you do not specify explicitly are copied from the appropriate default object. For example, if you define a local queue, those attributes that you do not specify are taken from the default queue SYSTEM.DEFAULT.LOCAL.QUEUE.

Clients and servers

WebSphere MQ supports client-server configurations for WebSphere MQ applications.

A *WebSphere MQ client* is a part of the WebSphere MQ product that is installed on a machine to accept MQI calls from applications and pass them to an *MQI server* machine. There they are processed by a queue manager. Typically, the client and server reside on different machines but they can also exist on the same machine.

Note: WebSphere MQ for i5/OS acts as a Java™ client only.

An *MQI server* is a queue manager that provides queuing services to one or more clients. All the WebSphere MQ objects, for example queues, exist only on the queue manager machine, that is, on the MQI server machine. A server can support normal local WebSphere MQ applications as well.

For more information about creating channels for clients and servers, see WebSphere MQ Intercommunication.

For information about client support in general, see WebSphere MQ Clients.

WebSphere MQ applications in a client-server environment

When linked to a server, client WebSphere MQ applications can issue most MQI calls in the same way as local applications. The client application issues an

MQCONN call to connect to a specified queue manager. Any additional MQI calls that specify the connection handle returned from the connect request are then processed by this queue manager.

The advantages of a client are that:

- It is simple to set up
- It is simple to manage
- It has a low resource footprint

You must link your applications to the appropriate client libraries. See WebSphere MQ Clients for further information.

Extending queue manager facilities

The facilities provided by a queue manager can be extended by defining user exits.

User exits

User exits provide a way for you to insert your own code into a queue manager function. The user exits supported include:

- **Channel exits**

  These exits change the way that channels operate. Channel exits are described in WebSphere MQ Intercommunication.
- **Data conversion exits**

  These exits create source code fragments that can be put into application programs to convert data from one format to another. Data conversion exits are described in the WebSphere MQ Application Programming Guide.
- **The cluster workload exit**

  The function performed by this exit is defined by the provider of the exit. Call definition information is given in WebSphere MQ Queue Manager Clusters. The exit is supported in the following environments: AIX, i5/OS, HP-UX, Solaris, Windows, and z/OS

Security

In WebSphere MQ for i5/OS security is provided by the Object Authority Manager (OAM) component. See Chapter 5, “Protecting WebSphere MQ objects,” on page 51 for details of this component.

Transactional support

An application program can group a set of updates into a *unit of work*. These updates are usually logically related and must all be successful for data integrity to be preserved. If one update succeeds while another fails, data integrity is lost.

A unit of work **commits** when it completes successfully. At this point all updates made within that unit of work are made permanent or irreversible. If the unit of work fails, all updates are instead *backed out*. *Syncpoint coordination* is the process by which units of work are either committed or backed out with integrity.

A *local* unit of work is one in which the only resources updated are those of the WebSphere MQ queue manager. Here, syncpoint coordination is provided by the queue manager itself using a dual-phase commit process and the MQI calls MQBACK and MQCMIT.

WebSphere MQ for i5/OS can support and participate in global units of work as a resource manager, coordinated by the i5/OS COMMIT and ROLLBACK commands.

Daylight saving time

In versions of WebSphere MQ earlier than Version 7.0, the time change caused problems for WebSphere MQ for i5/OS because WebSphere MQ used timestamps based on the system clock to access data in the queue manager's journal. In Websphere MQ Version 7.0 the timestamp includes additional information, so that log entries are unique, even if the system clock changes while WebSphere MQ is running.

Chapter 2. Managing WebSphere MQ for i5/OS using CL commands

WebSphere MQ for i5/OS provides three interfaces for administration:

- A set of fully featured i5/OS commands
- Provision for processing the WebSphere MQ script (MQSC) commands. This is a common interface used by all WebSphere MQ implementations on other platforms
- WebSphere MQ programmable command formats (PCF). This is a lower-level command interface, which can be used by applications to administer the product.

This chapter gives an overview of working with WebSphere MQ for i5/OS using the WebSphere MQ i5/OS commands, together with some suggested operations, as these commands are specific to i5/OS.

WebSphere MQ applications

When you create or customize WebSphere MQ applications, it is useful to keep a record of all WebSphere MQ definitions created. This record can be used for:

- Recovery purposes
- Maintenance
- Rolling out WebSphere MQ applications

You can do this by either:

- Creating CL programs to generate your WebSphere MQ definitions for the server, or
- Creating MQSC text files as SRC members to generate your WebSphere MQ definitions using the cross-platform WebSphere MQ command language.

  You are recommended to use SupportPac™ MS03 "WebSphere MQ - Save Queue Manager object definitions using PCFs" to generate the MQSC script file. You can download this SupportPac from the WebSphere MQ SupportPac Web page.

For further details about defining queue objects, see WebSphere MQ Script (MQSC) Command Reference or WebSphere MQ Programmable Command Formats and Administration Interface.

WebSphere MQ for i5/OS CL commands

The commands can be grouped as follows:

- Authentication Information Commands
  - CHGMQMAUTI, Change WebSphere MQ Authentication Information
  - CPYMQMAUTI, Copy WebSphere MQ Authentication Information
  - CRTMQMAUTI, Create WebSphere MQ Authentication Information
  - DLTMQMAUTI, Delete WebSphere MQ Authentication Information
  - DSPMQMAUTI, Display WebSphere MQ Authentication Information
  - WRKMQMAUTI, Work with WebSphere MQ Authentication Information

- Authority Commands
  - DSPMQMAUT, Display WebSphere MQ Object Authority
  - GRTMQMAUT , Grant WebSphere MQ Object Authority
  - RFRMQMAUT, Refresh WebSphere MQ Object Authority
  - RVKMQMAUT, Revoke WebSphere MQ Object Authority
  - WRKMQMAUT, Work with WebSphere MQ Authority
  - WRKMQMAUTD, Work with WebSphere MQ Authority Data
- Broker Commands
  - CLRMQMBRK, Clear WebSphere MQ Broker
  - DLTMQMBRK, Delete WebSphere MQ Broker
  - DSPMQMBRK, Display WebSphere MQ Broker
  - ENDMQMBRK, End WebSphere MQ Broker
  - STRMQMBRK, Start WebSphere MQ Broker
- Channel Commands
  - CHGMQMCHL, Change WebSphere MQ Channel
  - CPYMQMCHL, Copy WebSphere MQ Channel
  - CRTMQMCHL, Create WebSphere MQ Channel
  - DLTMQMCHL, Delete WebSphere MQ Channel
  - DSPMQMCHL, Display WebSphere MQ Channel
  - ENDMQMCHL, End WebSphere MQ Channel
  - PNGMQMCHL, Ping WebSphere MQ Channel
  - RSTMQMCHL, Reset WebSphere MQ Channel
  - RSVMQMCHL, Resolve WebSphere MQ Channel
  - STRMQMCHL, Start WebSphere MQ Channel
  - STRMQMCHLI, Start WebSphere MQ Channel Initiator
  - WRKMQMCHL, Work with WebSphere MQ Channels
  - WRKMQMCHST, Work with WebSphere MQ Channel Status
- Cluster Commands
  - RFRMQMCL, Refresh WebSphere MQ Cluster
  - RSMMQMCLQM, Resume WebSphere MQ Cluster Queue Manager
  - RSTMQMCL, Reset WebSphere MQ Cluster
  - SPDMQMCLQM, Suspend WebSphere MQ Cluster Queue Manager
  - WRKMQMCL, Work with WebSphere MQ Clusters
  - WRKMQMCLQ, Work with WebSphere MQ Cluster Queues
- Command Server Commands
  - DSPMQMCSVR, Display WebSphere MQ Command Server
  - ENDMQMCSVR, End WebSphere MQ Command Server
  - STRMQMCSVR, Start WebSphere MQ Command Server
- Connection Commands
  - ENDMQMCONN, End WebSphere MQ Connection
  - WRKMQMCONN, Work with WebSphere MQ Connections
- Data Conversion Exit Command
  - CVTMQMDTA, Convert WebSphere MQ Data Type
- Listener Commands
  - CHGMQMLSR, Change WebSphere MQ Listener Object

  - CPYMQMLSR, Copy WebSphere MQ Listener Object
  - CRTMQMLSR, Create WebSphere MQ Listener Object
  - DLTMQMLSR, Delete WebSphere MQ Listener Object
  - DSPMQMLSR, Display WebSphere MQ Listener Object
  - ENDMQMLSR, End WebSphere MQ Listener
  - STRMQMLSR, Start WebSphere MQ Listener
  - WRKMQMLSR, Work with WebSphere MQ Listeners
- Media Recovery Commands
  - RCDMQMIMG, Record WebSphere MQ Object Image
  - RCRMQMOBJ, Recreate WebSphere MQ Object
  - WRKMQMTRN, Work with WebSphere MQ Transactions
- Name Command
  - DSPMQMOBJN, Display WebSphere MQ Object Names
- Namelist Commands
  - CHGMQMNL, Change WebSphere MQ Namelist
  - CPYMQMNL, Copy WebSphere MQ Namelist
  - CRTMQMNL, Create WebSphere MQ Namelist
  - DLTMQMNL, Delete WebSphere MQ Namelist
  - DSPMQMNL, Display WebSphere MQ Namelist
  - WRKMQMNL, Work with WebSphere MQ Namelists
- Process Commands
  - CHGMQMPRC, Change WebSphere MQ Process
  - CPYMQMPRC, Copy WebSphere MQ Process
  - CRTMQMPRC, Create WebSphere MQ Process
  - DLTMQMPRC, Delete WebSphere MQ Process
  - DSPMQMPRC, Display WebSphere MQ Process
  - WRKMQMPRC, Work with WebSphere MQ Processes
- Queue Commands
  - CHGMQMQ, Change WebSphere MQ Queue
  - CLRMQMQ, Clear WebSphere MQ Queue
  - CPYMQMQ, Copy WebSphere MQ Queue
  - CRTMQMQ, Create WebSphere MQ Queue
  - DLTMQMQ, Delete WebSphere MQ Queue
  - DSPMQMQ, Display WebSphere MQ Queue
  - WRKMQMMSG, Work with WebSphere MQ Messages
  - WRKMQMQ, Work with WebSphere MQ Queues
  - WRKMQMQSTS, Work with WebSphere MQ Queue Status
- Queue Manager Commands
  - CCTMQM, Connect to Message Queue Manager
  - CHGMQM, Change Message Queue Manager
  - CRTMQM, Create Message Queue Manager
  - DLTMQM, Delete Message Queue Manager
  - DSCMQM, Disconnect from Message Queue Manager
  - DSPMQM, Display Message Queue Manager
  - DSPMQMSTS, Display Message Queue Manager Status

  - ENDMQM, End Message Queue Manager
  - STRMQM, Start Message Queue Manager
  - STRMQMTRM, Start WebSphere MQ Trigger Monitor
  - WRKMQM, Work with Message Queue Manager
- Service Commands
  - CHGMQMSVC, Change WebSphere MQ Service
  - CPYMQMSVC, Copy WebSphere MQ Service
  - CRTMQMSVC, Create WebSphere MQ Service
  - DLTMQMSVC, Delete WebSphere MQ Service
  - DSPMQMSVC, Display WebSphere MQ Service
  - ENDMQMSVC, End WebSphere MQ Service
  - STRMQMSVC, Start WebSphere MQ Service
  - WRKMQMSVC, Work with WebSphere MQ Services
- Subscription Commands
  - CHGMQMSUB, Change WebSphere MQ Subscription
  - CPYMQMSUB, Copy WebSphere MQ Subscription
  - CRTMQMSUB, Create WebSphere MQ Subscription
  - DLTMQMSUB, Delete WebSphere MQ Subscription
  - DSPMQMSUB, Display WebSphere MQ Subscription
  - WRKMQMSUB, Work with WebSphere MQ Subscription
- Topic Commands
  - CHGMQMTOP, Change WebSphere MQ Topic
  - CLRMQMTOP, Clear WebSphere MQ Topic
  - CPYMQMTOP, Copy WebSphere MQ Topic
  - CRTMQMTOP, Create WebSphere MQ Topic
  - DLTMQMTOP, Delete WebSphere MQ Topic
  - DSPMQMTOP, Display WebSphere MQ Topic
  - WRKMQMTOP, Work with WebSphere MQ Topics
- Trace Command
  - TRCMQM, Trace WebSphere MQ Job
- WebSphere MQSC Commands
  - RUNMQSC, Run WebSphere MQSC Commands
  - STRMQMMQSC, Start WebSphere MQSC Commands
- WebSphere MQ Dead-Letter Queue Handler Command
  - STRMQMDLQ, Start WebSphere MQ Dead-Letter Queue Handler
- WebSphere MQ Route Information
  - DSPMQMRTE, Display WebSphere MQ Route Information
- WebSphere MQ Version Details
  - DSPMQMVER, Display WebSphere MQ Version

General usage tips

Most groups of WebSphere MQ commands, including those associated with queue managers, queues, topics, channels, namelists, process definitions, and authentication information objects can be accessed using the relevant WRK* command.

The principal command in the set is WRKMQM. This command allows you, for example, to display a list of all the queue managers on the system, together with status information. Alternatively, you can process all queue-manager specific commands using various options against each entry.

From the WRKMQM command you can select specific areas of each queue manager, for example, working with channels, topics or queues, and from there select individual objects.

Before you start

Ensure that the WebSphere MQ subsystem is running (using the command `STRSBS QMQM/QMQM`), and that the job queue associated with that subsystem is not held. By default, the WebSphere MQ subsystem and job queue are both named QMQM in library QMQM.

Starting a local queue manager

Using the i5/OS command line to start a queue manager

You must:

1. Create a local queue manager by issuing the **CRTMQM** command from an i5/OS command line.

   When you create a queue manager, you have the option of making that queue manager the default queue manager.

   The default queue manager (of which there can only be one) is the queue manager to which a CL command applies, if the queue manager name (MQMNAME) parameter is omitted.
2. Start a local queue manager by issuing the **STRMQM** command from an i5/OS command line.

   If the queue manager start up takes more than a few seconds WebSphere MQ will show status messages intermittently detailing the start up progress. For more information on these messages see WebSphere MQ Messages.

You can stop a queue manager by issuing the **ENDMQM** command from the i5/OS command line, and control a queue manager by issuing other WebSphere MQ commands from an i5/OS command line.

The principal commands are described later in this chapter.

Remote queue managers cannot be started remotely but must be created and started in their systems by local operators. An exception to this is where remote operating facilities (outside WebSphere MQ for i5/OS) exist to enable such operations.

The local queue administrator cannot stop a remote queue manager.

Note: As part of quiescing a WebSphere MQ (or MQSeries®) system, you have to quiesce the active queue managers. This is described in Chapter 14, “Quiescing WebSphere MQ for i5/OS,” on page 691.

Creating WebSphere MQ objects

The following tasks suggest various ways in which you can use WebSphere MQ for i5/OS from the command line.

There are two online methods to create WebSphere MQ objects, which are:

1. Using a Create command, for example:

   CRTMQMQ
   Create MQM Queue

2. Using a Work with MQM object command, followed by F6, for example:

   WRKMQMQ
   Work with MQM Queues

For a list of all commands see Chapter 11, "The CL commands," on page 197.

Note: All MQM commands can be submitted from the Message Queue Manager Commands menu. To display this menu, type `GO CMDMQM` on the command line and press the `Enter` key.

The system displays the prompt panel automatically when you select a command from this menu. To display the prompt panel for a command that you have typed directly on the command line, press `F4` before pressing the `Enter` key.

Examples of creating a local queue

To create a local queue from the command line, use:

1. The Create MQM Queue (**CRTMQMQ**) command
2. The Work with MQM Queues (**WRKMQMQ**) command, followed by F6

Creating a local queue using the CRTMQMQ command

1. Type **CRTMQMQ** on the command line and press the `F4` key.
2. On the Create MQM Queue panel, type the name of the queue that you want to create in the `Queue name` field.

   To specify a mixed case name, you enclose the name in apostrophes.
3. Type `*LCL` in the `Queue type` field.
4. Specify a queue manager name, unless you are using the default queue manager, and press the `Enter` key. Further settings for a local queue are displayed (see Figure Figure 1 on page 21) with the fields containing the default values. You can overtype any of these values with a new value.

   Scroll forward to see further fields. The options used for clusters are at the end of the list of options.
5. When you have changed any values, press the `Enter` key to create the queue.

```
                        Create MQM Queue (CRTMQMQ)

Type choices, press Enter.

Queue name . . . . . . . . . . . > TEST.QUEUE.LCL

Queue type . . . . . . . . . . . > *LCL          *ALS, *LCL, *MDL, *RMT
Message Queue Manager Name . . .   MY.QUEUE.MANAGER_____________________
_____________________________
Replace  . . . . . . . . . . . .   *NO_          *NO, *YES
Text 'description' . . . . . . .   '_______________________________________
_____________________________'
Put enabled  . . . . . . . . . .   *YES____      *SYSDFTQ, *NO, *YES
Default message priority . . . .   5_________    0-9, *SYSDFTQ
Default message persistence  . .   *NO_____      *SYSDFTQ, *NO, *YES
Process name . . . . . . . . . .   '_______________________________________
_____'
Triggering enabled . . . . . . .   *NO_____      *SYSDFTQ, *NO, *YES
Get enabled  . . . . . . . . . .   *YES____      *SYSDFTQ, *NO, *YES
Sharing enabled  . . . . . . . .   *YES____      *SYSDFTQ, *NO, *YES
                                                                  More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 1. Create MQM Queue initial panel

Creating a local queue using the WRKMQMQ command

1. Type **WRKMQMQ** on the command line.
2. Enter the name of a queue manager
3. If you want to display the prompt panel, press F4.

   The prompt panel is useful to reduce the number of queues displayed, by specifying a generic queue name or queue type.
4. Press the Enter key and Figure Figure 2 is displayed.

```
                           Work with MQ Queues

Queue Manager Name . . :   common

Type options, press Enter.
  2=Change    3=Copy    4=Delete   5=Display   8=Work status
  12=Work with messages 13=Clear...


Opt  Name                                          Type  Depth   Jobs
     SYSTEM.ADMIN.CHANNEL.EVENT                    *LCL      0      0
     SYSTEM.ADMIN.COMMAND.QUEUE                    *LCL      0      0
     SYSTEM.ADMIN.PERFM.EVENT                      *LCL      0      0
     SYSTEM.ADMIN.QMGR.EVENT                       *LCL      1      0
     SYSTEM.AUTH.DATA.QUEUE                        *LCL     29      1
     SYSTEM.CHANNEL.INITQ                          *LCL      0      1
     SYSTEM.CHANNEL.SYNCQ                          *LCL      1      0
     SYSTEM.CICS.INITIATION.QUEUE                  *LCL      0      0
     SYSTEM.CLUSTER.COMMAND.QUEUE                  *LCL      0      1
                                                                  More...
Parameters for options 2, 3, 4, 5, 12, 13, 14, 15, 16 or command
===>
F3=Exit   F4=Prompt   F5=Refresh   F6=Create   F7=Filter       F9=Retrieve
F11=Change View   F12=Cancel    F16=Repeat find   F17=Find   F24=More keys
```

Figure 2. Work with MQM Queues panel

5. Press F6 to create a new queue; this takes you to the **CRTMQMQ** panel. See "Creating a local queue using the CRTMQMQ command" on page 20 for instructions on how to create the queue.

When you have created the queue, the Work with MQM Queues panel is displayed again. The new queue is added to the list when you press F5=Refresh.

Examples of creating a remote queue

Use the CRTMQMQ panel to define the queue with queue type *RMT, using one of the following online methods:

1. The **CRTMQMQ** command.
2. F6=Create on the **WRKMQMQ** panel.

The use of remote queues is described in detail in WebSphere MQ Intercommunication.

This section describes how to define a remote queue for each of the three uses. We use the CRTMQMQ command in the examples; you can, of course, do the same thing from the WRKMQMQ panel.

Creating a remote queue as a remote queue definition

This is the most straightforward use of remote queues. It is used to direct messages to a local queue on a remote queue manager, through a transmission queue.

To create a remote queue for this use:

1. Type **CRTMQMQ** on the command line and press the F4 key.
2. Type the queue name in the Queue name field.
3. Type *RMT in the Queue type field.
4. Type the name of the local queue manager in the Queue Manager Name field.
5. Type the name of the local queue at the remote location in the Remote queue field.
6. Type the name of the queue manager at the remote location in the Remote Message Queue Manager field.
7. Optionally, type the name of the transmission queue to the remote location in the Transmission queue field.

   If you do not specify a transmission queue name, the transmission queue with the same name as the remote queue manager is used.

Creating a remote queue as a queue manager alias

Queue manager alias definitions can be used to remap the queue manager name specified in the MQOPEN call. This enables you to alter the target queue manager without changing your applications.

See WebSphere MQ Intercommunication for further information.

To define a remote queue as a queue manager object:

1. Type **CRTMQMQ** on the command line and press the F4 key.
2. Type the queue name in the Queue name field.
3. Type *RMT in the Queue type field.
4. Type the name of the local queue manager in the Queue Manager Name field.
5. Type the name of the queue manager at the remote location in the Remote Message Queue Manager field.

6. Optionally, type the name of the transmission queue to the remote location in the Transmission queue field.

   If you do not specify a transmission queue name, the transmission queue with the same name as the remote queue manager is used.

Creating a remote queue as an alias to a reply-to queue

An application can name a reply-to queue when it puts a message on a queue. The reply-to queue name is used by the application that gets the message from the queue to send reply messages. To define an alias to a reply-to queue, define a remote queue with the same name as the reply-to queue.

See WebSphere MQ Intercommunication for further information.

To create a remote queue as an alias to a reply-to queue:

1. Type **CRTMQMQ** on the command line and press the F4 key.
2. Type the queue name in the Queue name field.

   This must be the same as the reply-to queue named by the putting application.
3. Type *RMT in the Queue type field.
4. Type the name of the local queue manager in the Queue Manager Name field, unless you are using the default queue manager.
5. Type the queue name in the Queue name field.

   This is the name of the queue to which you want the reply-to messages sent.
6. Type the name of the queue manager at the remote location in the Remote Message Queue Manager field.

   This is the name of the queue manager to which you want the reply-to messages sent.
7. Optionally, type the name of the transmission queue to the remote location in the Transmission queue field.

   If you do not specify a transmission queue name, the transmission queue with the same name as the remote queue manager is used.

Creating a transmission queue

A transmission queue is a local queue that is used to send messages to a remote queue manager, through a message channel, which provides a one-way link to the remote queue manager.

Each message channel has a transmission queue name specified at the sending end of the message channel.

Note: If you use clusters, you do not have to create a transmission queue.

To create a transmission queue:

1. Type **CRTMQMQ** on the command line and press the F4 key.
2. Type the queue name in the Queue name field.

   If you want to define a default transmission queue for all messages destined to a remote queue manager, the transmission queue name must be the same as the remote queue manager name.
3. Type *LCL in the Queue type field.
4. Type *TMQ in the Usage field.

Creating an initiation queue

An initiation queue is a local queue on which the queue manager puts trigger messages in response to a trigger event, for example, a message arriving on a local queue. An initiation queue is a local queue and has no special settings that define it as an initiation queue.

For more information about triggering, see the WebSphere MQ Application Programming Guide.

Creating an alias queue

Use an alias queue object to access another queue on the local queue manager. Any messages put on the alias queue are redirected to the queue named in the alias queue definition.

Note: An alias queue cannot hold messages itself.

To create an alias queue:

1. Type **CRTMQMQ** on the command line and press the `F4` key.
2. Type the queue name in the `Queue name` field.
3. Type `*ALS` in the `Queue type` field.
4. Type the name of the local queue manager in the `Queue Manager Name` field.
5. Type the name of the local queue that you want the queue name to resolve to in the `Target queue` field.

Creating a model queue

Define a model queue with a set of attributes in the same way that you define a local queue. Type `*MDL` in the `Queue type` field.

Model queues and local queues have the same set of attributes, except that on model queues you can specify whether the dynamic queues created are temporary or permanent. (Permanent queues are maintained across queue manager restarts, temporary ones are not.)

Altering queue manager attributes

To alter the attributes of the queue manager specified on the **CHGMQM** command, specifying the attributes and values that you want to change. For example, use the following options to alter the attributes of `jupiter.queue.manager`:

```
CHGMQM MQMNAME('jupiter.queue.manager') UDLMSGQ(ANOTHERDLQ) INHEVT(*YES)
```

This command changes the dead-letter queue used, and enables inhibit events.

Working with local queues

This section contains examples of some of the commands that you can use to manage local, model, and alias queues. All the commands shown are also available using options from the WRKMQMQ command panel.

Defining a local queue

For an application, the local queue manager is the queue manager to which the application is connected. Queues that are managed by the local queue manager are said to be local to that queue manager.

Use the command CRTMQMQ QTYPE *LCL to create a definition of a local queue and also to create the data structure that is called a queue. You can also modify the queue characteristics from those of the default local queue.

In this example, the queue we define, ORANGE.LOCAL.QUEUE, is specified to have these characteristics:

- It is enabled for gets, disabled for puts, and operates on a first-in-first-out (FIFO) basis.
- It is an *ordinary* queue, that is, it is not an initiation queue or a transmission queue, and it does not generate trigger messages.
- The maximum queue depth is 1000 messages; the maximum message length is 2000 bytes.

The following command does this on the default queue manager:

```
CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL)
       TEXT('Queue for messages from other systems')
       PUTENBL(*NO)
       GETENBL(*YES)
       TRGENBL(*NO)
       MSGDLYSEQ(*FIFO)
       MAXDEPTH(1000)
       MAXMSGLEN(2000)
       USAGE(*NORMAL)
```

Note:

1. USAGE *NORMAL indicates that this queue is not a transmission queue.
2. If you already have a local queue on the same queue manager with the name orange.local.queue, this command fails. Use the REPLACE *YES attribute, if you want to overwrite the existing definition of a queue, but see also "Changing local queue attributes" on page 26.

Defining a dead-letter queue

Each queue manager must have a local queue to be used as a dead-letter queue so that messages that cannot be delivered to their correct destination can be stored for later retrieval. You must explicitly tell the queue manager about the dead-letter queue. You can do this by specifying a dead-letter queue on the **CRTMQM** command, or you can use the **CHGMQM** command to specify one later. You must also define the dead-letter queue before it can be used.

A sample dead-letter queue called SYSTEM.DEAD.LETTER.QUEUE is supplied with the product. This queue is automatically created when you create the queue manager. You can modify this definition if required. There is no need to rename it, although you can if you like.

A dead-letter queue has no special requirements except that:

- It must be a local queue

- Its MAXMSGL (maximum message length) attribute must enable the queue to accommodate the largest messages that the queue manager has to handle **plus** the size of the dead-letter header (MQDLH)

WebSphere MQ provides a dead-letter queue handler that allows you to specify how messages found on a dead-letter queue are to be processed or removed. For further information, see Chapter 6, "The WebSphere MQ dead-letter queue handler," on page 83.

Displaying default object attributes

When you define a WebSphere MQ object, it takes any attributes that you do not specify from the default object. For example, when you define a local queue, the queue inherits any attributes that you omit in the definition from the default local queue, which is called SYSTEM.DEFAULT.LOCAL.QUEUE. To see exactly what these attributes are, use the following command:

```
DSPMQMQ QNAME(SYSTEM.DEFAULT.LOCAL.QUEUE) MQMNAME(MYQUEUEMANAGER)
```

Copying a local queue definition

You can copy a queue definition using the **CPYMQMQ** command. For example:

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

This command creates a queue with the same attributes as our original queue orange.local.queue, rather than those of the system default local queue.

You can also use the CPYMQMQ command to copy a queue definition, but substituting one or more changes to the attributes of the original. For example:

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('third.queue') MQMNAME(MYQUEUEMANAGER)
MAXMSGLEN(1024)
```

This command copies the attributes of the queue orange.local.queue to the queue third.queue, but specifies that the maximum message length on the new queue is to be 1024 bytes, rather than 2000.

Note: When you use the **CPYMQMQ** command, you copy the queue attributes only, not the messages on the queue.

Changing local queue attributes

You can change queue attributes in two ways, using either the **CHGMQMQ** command or the **CPYMQMQ** command with the REPLACE *YES attribute. In "Defining a local queue" on page 25, we defined the queue orange.local.queue. Suppose, for example, you wanted to increase the maximum message length on this queue to 10 000 bytes.

- Using the **CHGMQMQ** command:

  ```
  CHGMQMQ QNAME('orange.local.queue') MQMNAME(MYQUEUEMANAGER) MAXMSGLEN(10000)
  ```

  This command changes a single attribute, that of the maximum message length; all the other attributes remain the same.
- Using the **CRTMQMQ** command with the REPLACE *YES option, for example:

  ```
  CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL) MQMNAME(MYQUEUEMANAGER)
  MAXMSGLEN(10000) REPLACE(*YES)
  ```

This command changes not only the maximum message length, but all the other attributes, which are given their default values. The queue is now put enabled whereas previously it was put inhibited. Put enabled is the default, as specified by the queue SYSTEM.DEFAULT.LOCAL.QUEUE, unless you have changed it.

If you ***decrease*** the maximum message length on an existing queue, existing messages are not affected. Any new messages, however, must meet the new criteria.

Clearing a local queue

To delete all the messages from a local queue called magenta.queue, use the following command:

```
CLRMQMQ QNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

You cannot clear a queue if:

- There are uncommitted messages that have been put on the queue under syncpoint.
- An application currently has the queue open.

Deleting a local queue

Use the command DLTMQMQ to delete a local queue.

A queue cannot be deleted if it has uncommitted messages on it, or if it is in use.

Enabling large queues

WebSphere MQ supports queues larger than 2 GB. See your operating system documentation for information on how to enable i5/OS to support large files.

The i5/OS Information Center can be found here: http://publib.boulder.ibm.com/iseries/

Some utilities might not be able to cope with files greater than 2 GB. Before enabling large file support, check your operating system documentation for information on restrictions on such support.

Working with alias queues

An alias queue (sometimes known as a queue alias) provides a method of redirecting MQI calls. An alias queue is not a real queue but a definition that resolves to a real queue. The alias queue definition contains a target queue name, which is specified by the TGTQNAME attribute.

When an application specifies an alias queue in an MQI call, the queue manager resolves the real queue name at run time.

For example, an application has been developed to put messages on a queue called my.alias.queue. It specifies the name of this queue when it makes an MQOPEN request and, indirectly, if it puts a message on this queue. The application is not aware that the queue is an alias queue. For each MQI call using this alias, the queue manager resolves the real queue name, which could be either a local queue or a remote queue defined at this queue manager.

By changing the value of the TGTQNAME attribute, you can redirect MQI calls to another queue, possibly on another queue manager. This is useful for maintenance, migration, and load-balancing.

Defining an alias queue

The following command creates an alias queue:

```
CRTMQMQ QNAME('my.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
MQMNAME(MYQUEUEMANAGER)
```

This command redirects MQI calls that specify my.alias.queue to the queue yellow.queue. The command does not create the target queue; the MQI calls fail if the queue yellow.queue does not exist at run time.

If you change the alias definition, you can redirect the MQI calls to another queue. For example:

```
CHGMQMQ QNAME('my.alias.queue') TGTQNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

This command redirects MQI calls to another queue, magenta.queue.

You can also use alias queues to make a single queue (the target queue) appear to have different attributes for different applications. You do this by defining two aliases, one for each application. Suppose there are two applications:

- Application ALPHA can put messages on yellow.queue, but is not allowed to get messages from it.
- Application BETA can get messages from yellow.queue, but is not allowed to put messages on it.

You can do this using the following commands:

```
/* This alias is put enabled and get disabled for application ALPHA */

CRTMQMQ QNAME('alphas.alias.queue') QTYPE(*ALS)  TGTQNAME('yellow.queue')
PUTENBL(*YES) GETENBL(*NO) MQMNAME(MYQUEUEMANAGER)

/* This alias is put disabled and get enabled for application BETA */

CRTMQMQ QNAME('betas.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
PUTENBL(*NO) GETENBL(*YES) MQMNAME(MYQUEUEMANAGER)
```

ALPHA uses the queue name alphas.alias.queue in its MQI calls; BETA uses the queue name betas.alias.queue. They both access the same queue, but in different ways.

You can use the REPLACE *YES attribute when you define alias queues, in the same way that you use these attributes with local queues.

Using other commands with alias queues

You can use the appropriate commands to display or change alias queue attributes. For example:

```
 * Display the alias queue's attributes */

DSPMQMQ QNAME('alphas.alias.queue') MQMNAME(MYQUEUEMANAGER)

/* ALTER the base queue name, to which the alias resolves. */
```

```
/* FORCE = Force the change even if the queue is open. */

CHQMQMQ QNAME('alphas.alias.queue') TGTQNAME('orange.local.queue') FORCE(*YES)
 MQMNAME(MYQUEUEMANAGER)
```

Working with model queues

A queue manager creates a *dynamic queue* if it receives an MQI call from an application specifying a queue name that has been defined as a model queue. The name of the new dynamic queue is generated by the queue manager when the queue is created. A *model queue* is a template that specifies the attributes of any dynamic queues created from it.

Model queues provide a convenient method for applications to create queues as they are required.

Defining a model queue

You define a model queue with a set of attributes in the same way that you define a local queue. Model queues and local queues have the same set of attributes, except that on model queues you can specify whether the dynamic queues created are temporary or permanent. (Permanent queues are maintained across queue manager restarts, temporary ones are not). For example:

```
CRTMQMQ QNAME('green.model.queue') QTYPE(*MDL) DFNTYPE(*PERMDYN)
```

This command creates a model queue definition. From the DFNTYPE attribute, the actual queues created from this template are permanent dynamic queues. The attributes not specified are automatically copied from the SYSYTEM.DEFAULT.MODEL.QUEUE default queue.

You can use the REPLACE *YES attribute when you define model queues, in the same way that you use them with local queues.

Using other commands with model queues

You can use the appropriate commands to display or alter a model queue's attributes. For example:

```
/* Display the model queue's attributes */

DSPMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('green.model.queue')

/* ALTER the model queue to enable puts on any */
/* dynamic queue created from this model. */

CHGMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('blue.model.queue') PUTENBL(*YES)
```

Working with triggering

WebSphere MQ provides a facility for starting an application automatically when certain conditions on a queue are met. One example of the conditions is when the number of messages on a queue reaches a specified number. This facility is called *triggering* and is described in detail in the WebSphere MQ Application Programming Guide.

What is triggering?

The queue manager defines certain conditions as constituting trigger events. If triggering is enabled for a queue and a trigger event occurs, the queue manager sends a trigger message to a queue called an initiation queue. The presence of the trigger message on the initiation queue indicates that a trigger event has occurred.

Trigger messages generated by the queue manager are not persistent. This has the effect of reducing logging (thereby improving performance), and minimizing duplicates during restart, so improving restart time.

What is the trigger monitor?

The program which processes the initiation queue is called a trigger-monitor application, and its function is to read the trigger message and take appropriate action, based on the information contained in the trigger message. Normally this action would be to start some other application to process the queue which caused the trigger message to be generated. From the point of view of the queue manager, there is nothing special about the trigger-monitor application--it is simply another application that reads messages from a queue (the initiation queue).

Altering the job submission attributes of the trigger monitor

The trigger monitor supplied as command STRMQMTRM submits a job for each trigger message using the system default job description, QDFTJOBD. This has limitations in that the submitted jobs are always called QDFTJOBD and have the attributes of the default job description including the library list, *SYSVAL. WebSphere MQ provides a method for overriding these attributes. For example, it is possible to tailor the submitted jobs to have more meaningful job names as follows:

1. In the job description specify the description you want, for example logging values.
2. Specify the Environment Data of the process definition used in the triggering process:

   ```
   CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOBD(MYLIB/TRIGJOBD)')
   ```

   The Trigger Monitor now performs a SBMJOB using the specified description.

```
              Display MQ Process


Queue manager name . . . . . > QmgrName

Process name . . . . . > ProcessName

Text 'description' . . . . . >

Application type . . . . . > *OS400
Application identifier . . . > LibName/PgmName

User data  . . . . . >

Environment data  . . . . . > USER(UserName) JOBNAME(JobName)
JOBD(*LIBL/QDFTJOBD) LOG( 4 00 *Seclvl)

Last alter date . . . . . > 2005-05-01
Last alter time . . . . . > 11.44.29

F3=Exit   F12=Cancel   F21=Print
```

Figure 3. Display MQ Process panel

It is possible to override other attributes of the SBMJOB by specifying the appropriate keyword and value in the Environment Data of the process definition. The only exception to this is the CMD keyword because this attribute is filled by the trigger monitor. An example of the command to specify the Environment Data of the process definition where both the job name and description are to be altered follows:

```
CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOBD(MYLIB/TRIGJOB)
   JOB(TRIGGER)')
```

Setting up objects for triggering

This following sections describe how to set up the required objects to support triggering on WebSphere MQ.

Defining an application queue for triggering

An application queue is a local queue that is used by applications for messaging, through the MQI. Triggering requires a number of queue attributes to be defined on the application queue. Triggering itself is enabled by the TRGENBL attribute.

In this example, a trigger event is to be generated when there are 100 messages of priority 5 or higher on the local queue motor.insurance.queue, as follows:

```
CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.insurance.queue') QTYPE(*LCL)
       PRCNAME('motor.insurance.quote.process') MAXMSGLEN(2000)
       DFTMSGPST(*YES) INITQNAME('motor.ins.init.queue')
       TRGENBL(*YES) TRGTYPE(*DEPTH) TRGDEPTH(100) TRGMSGPTY(5)
```

where the parameters are:

MQMNAME(MYQUEUEMANAGER)
The name of the queue manager.

QNAME('motor.insurance.queue')
The name of the application queue being defined.

PRCNAME('motor.insurance.quote.process')
The name of the application to be started by a trigger monitor program.

MAXMSGLEN(2000)
The maximum length of messages on the queue.

DFTMSGPST(*YES)
Messages on this queue are persistent by default.

INITQNAME('motor.ins.init.queue')
The name of the initiation queue on which the queue manager is to put the trigger message.

TRGENBL(*YES)
The trigger attribute value.

TRGTYPE(*DEPTH)
A trigger event is generated when the number of messages of the required priority (TRGMSGPTY) reaches the number specified in TRGDEPTH.

TRGDEPTH(100)
The number of messages required to generate a trigger event.

TRGMSGPTY(5)
The priority of messages that are to be counted by the queue manager in deciding whether to generate a trigger event. Only messages with priority 5 or higher are counted.

Defining an initiation queue

When a trigger event occurs, the queue manager puts a trigger message on the initiation queue specified in the application queue definition. Initiation queues have no special settings, but you can use the following definition of the local queue motor.ins.init.queue for guidance:

```
CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.ins.init.queue') QTYPE(*LCL)
       GETENBL(*YES) SHARE(*NO) TRGTYPE(*NONE)
       MAXMSGL(2000)
       MAXDEPTH(1000)
```

Creating a process definition

Use the CRTMQMPRC command to create a process definition. A process definition associates an application queue with the application that is to process messages from the queue. This is done through the PRCNAME attribute on the application queue motor.insurance.queue. The following command creates the required process, motor.insurance.quote.process, identified in this example:

```
CRTMQMPRC MQMNAME(MYQUEUEMANAGER) PRCNAME('motor.insurance.quote.process')
          TEXT('Insurance request message processing')
          APPTYPE(*OS400) APPID(MQTEST/TESTPROG)
          USRDATA('open, close, 235')
```

where the parameters are:

MQMNAME(MYQUEUEMANAGER)
The name of the queue manager.

PRCNAME('motor.insurance.quote.process')
The name of the process definition.

TEXT('Insurance request message processing')
A description of the application program to which this definition relates. This text is displayed when you use the DSPMQMPRC command. This can help you to identify what the process does. If you use spaces in the string, you must enclose the string in single quotation marks.

APPTYPE(*OS400)
The type of application to be started.

APPID(MQTEST/TESTPROG)
The name of the application executable file, specified as a fully qualified file name.

USRDATA('open, close, 235')
User-defined data, which can be used by the application.

Displaying your process definition

Use the **DSPMQMPRC** command to examine the results of your definition. For example:

```
MQMNAME(MYQUEUEMANAGER) DSPMQMPRC('motor.insurance.quote.process')
```

You can also use the **CHGMQMPRC** command to alter an existing process definition, and the **DLTMQMPRC** command to delete a process definition.

Communicating between two systems

The following example illustrates how to set up two WebSphere MQ for i5/OS systems, using CL commands, so that they can communicate with one another.

The systems are called SYSTEMA and SYSTEMB, and the communications protocol used is TCP/IP.

Carry out the following procedure:

1. Create a queue manager on SYSTEMA, calling it QMGRA1.

```
CRTMQM     MQMNAME(QMGRA1) TEXT('System A - Queue +
             Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

2. Start this queue manager.

```
STRMQM     MQMNAME(QMGRA1)
```

3. Define the WebSphere MQ objects on SYSTEMA that you need to send messages to a queue manager on SYSTEMB.

```
/* Transmission queue */
 CRTMQMQ    QNAME(XMITQ.TO.QMGRB1) QTYPE(*LCL) +
              MQMNAME(QMGRA1) TEXT('Transmission Queue +
              to QMGRB1') MAXDEPTH(5000) USAGE(*TMQ)

/* Remote queue that points to a queue called TARGETB               */
/* TARGETB belongs to queue manager QMGRB1 on SYSTEMB               */
 CRTMQMQ    QNAME(TARGETB.ON.QMGRB1) QTYPE(*RMT) +
              MQMNAME(QMGRA1) TEXT('Remote Q pointing +
              at Q TARGETB on QMGRB1 on Remote System +
              SYSTEMB') RMTQNAME(TARGETB) +
              RMTMQMNAME(QMGRB1) TMQNAME(XMITQ.TO.QMGRB1)

/* TCP/IP sender channel to send messages to the queue manager on SYSTEMB*/
 CRTMQMCHL  CHLNAME(QMGRA1.TO.QMGRB1) CHLTYPE(*SDR) +
              MQMNAME(QMGRA1) TRPTYPE(*TCP) +
              TEXT('Sender Channel From QMGRA1 on +
              SYSTEMA to QMGRB1 on SYSTEMB') +
              CONNAME(SYSTEMB) TMQNAME(XMITQ.TO.QMGRB1)
```

4. Create a queue manager on SYSTEMB, calling it QMGRB1.

```
CRTMQM     MQMNAME(QMGRB1) TEXT('System B  - Queue +
             Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

5. Start the queue manager on SYSTEMB.

```
STRMQM     MQMNAME(QMGRB1)
```

6. Define the WebSphere MQ objects that you need to receive messages from the queue manager on SYSTEMA.

```
/* Local queue to receive messages on */
 CRTMQMQ    QNAME(TARGETB) QTYPE(*LCL) MQMNAME(QMGRB1) +
              TEXT('Sample Local Queue for QMGRB1')

/* Receiver channel of the same name as the sender channel on SYSTEMA */
 CRTMQMCHL  CHLNAME(QMGRA1.TO.QMGRB1) CHLTYPE(*RCVR) +
              MQMNAME(QMGRB1) TRPTYPE(*TCP) +
              TEXT('Receiver Channel from QMGRA1 to +
              QMGRB1')
```

7. Finally, start a TCP/IP listener on SYSTEMB so that the channel can be started. This example uses the default port of 1414.

```
STRMQMLSR MQMNAME(QMGRB1)
```

You are now ready to send test messages between SYSTEMA and SYSTEMB. Using one of the supplied samples, put a series of messages to your remote queue on SYSTEMA.

Start the channel on SYSTEMA, either by using the command **STRMQMCHL**, or by using the command **WRKMQMCHL** and entering a start request (Option 14) against the sender channel.

The channel should go to RUNNING status and the messages are sent to queue TARGETB on SYSTEMB.

Check your messages by issuing the command:

```
WRKMQMMSG QNAME(TARGETB) MQMNAME(QMGRB1).
```

Chapter 3. Alternative ways of administering WebSphere MQ

You normally use i5/OS CL commands to administer WebSphere MQ for i5/OS. See Chapter 2, "Managing WebSphere MQ for i5/OS using CL commands," on page 15 for an overview of these commands.

Using CL commands is the preferred method of administering the system. However, you can use various other methods. This chapter gives an overview of those methods and includes the following topics:

- "Local and remote administration"
- "Administration using MQSC commands" on page 36
- "Administration using PCF commands" on page 37
- "Using the WebSphere MQ Explorer with WebSphere MQ for i5/OS" on page 39
- "Managing the command server for remote administration" on page 40

Local and remote administration

You administer WebSphere MQ objects locally or remotely.

Local administration means carrying out administration tasks on any queue managers that you have defined on your local system. In WebSphere MQ, you can consider this as local administration because no WebSphere MQ channels are involved, that is, the communication is managed by the operating system. Some commands cannot be issued in this way, in particular, creating or starting queue managers and starting command servers. To perform this type of task, you must either log onto the remote system and issue the commands from there, or create a process that can issue the commands for you.

WebSphere MQ supports administration from a single point through what is known as *remote administration*. Remote administration consists of sending programmable command format (PCF) control messages to the SYSTEM.ADMIN.COMMAND.QUEUE on the target queue manager.

There are a number of ways of generating PCF messages. These are:

1. Writing a program using PCF messages. See "Administration using PCF commands" on page 37.
2. Writing a program using the MQAI, which sends out PCF messages. See "Using the MQAI to simplify the use of PCFs" on page 38.
3. Using the WebSphere MQ Explorer, available with WebSphere MQ for Windows, which allows you to use a graphical user interface (GUI) and generates the correct PCF messages. See "Using the WebSphere MQ Explorer with WebSphere MQ for i5/OS" on page 39.

For example, you can issue a remote command to change a queue definition on a remote queue manager.

Some commands cannot be issued in this way, in particular, creating or starting queue managers and starting command servers. To perform this type of task, you must either log onto the remote system and issue the commands from there or create a process that can issue the commands for you.

Administration using MQSC commands

You use WebSphere MQ script (MQSC) commands to manage queue manager objects, including the queue manager itself, queues, process definitions, namelists, channels, client connection channels, listeners, services, and authentication information objects.

You issue MQSC commands to a queue manager using the STRMQMMQSC WebSphere MQ CL command. This is a batch method only, taking its input from a source physical file in the server library system. The default name for this source physical file is QMQSC.

WebSphere MQ for i5/OS does not supply a source file called QMQSC. To process MQSC commands you need to create the QMQSC source file in a library of your choice, by issuing the following command:

```
CRTSRCPRF FILE(MYLIB/QMQSC) TEXT('WebSphere MQ - MQSC Source')
```

MQSC source is held in members within this source file. To work with the members enter the following command:

```
 WRKMBRPDM MYLIB/QMQSC
```

You can now add new members and maintain existing ones

You can also enter MQSC commands interactively, by issuing RUNMQSC or:

1. Typing in the queue manager name and pressing the `Enter` key to access the WRKMQM results panel
2. Selecting `F23=More options` on this panel
3. Selecting option 26 against an active queue manager on the panel shown in Figure 4

To end such an MQSC session, type **end**.

```
                          Work with queue managers

Type options, press Enter.
  21=Work with NameLists     22=Work with Jobs       23=Display logs
  24=Work with Authorities   25=Work with AuthInfo   26=MQSC ...

Opt  Name                                                  Status  Default
     MICK                                                  ACTIVE     NO
     MIKE                                                  ACTIVE     NO









                                                                      Bottom


Parameters or command
===>
F3=Exit   F4=Prompt      F5=Refresh   F6=Create   F9=Retrieve   F12=Cancel
F16=Repeat position to   F17=Position to   F23=More options     F24=More keys
```

Figure 4. Work with queue managers results panel

MQSC command files

MQSC commands are written in human-readable form, that is, in EBCDIC text.

Figure 5 is an extract from an MQSC command file showing an MQSC command (DEFINE QLOCAL) with its attributes.

```
 .
 .
DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) REPLACE  +
       DESCR(' ') +
       PUT(ENABLED) +
       DEFPRTY(0) +
       DEFPSIST(NO) +
       GET(ENABLED) +
       MAXDEPTH(5000) +
       MAXMSGL(1024) +
       DEFSOPT(SHARED) +
       NOHARDENBO +
       USAGE(NORMAL) +
       NOTRIGGER;
 .
 .
```

Figure 5. Extract from the MQSC command file, myprog.in

For portability among WebSphere MQ environments, limit the line length in MQSC command files to 72 characters. The plus sign indicates that the command is continued on the next line.

Object attributes specified in MQSC are shown in this book in uppercase (for example, RQMNAME), although they are not case sensitive.

Note:

1. The format of an MQSC file does not depend on its location in the file system
2. MQSC attribute names are limited to eight characters.
3. MQSC commands are available on other platforms, including z/OS.

The WebSphere MQ Script (MQSC) Command Reference contains a description of each MQSC command and its syntax.

Administration using PCF commands

The purpose of WebSphere MQ programmable command format (PCF) commands is to allow administration tasks to be programmed into an administration program. In this way you can create queues and process definitions, and change queue managers, from a program.

PCF commands cover the same range of functions provided by MQSC commands. However, unlike MQSC commands, PCF commands and their replies are not in a text format that you can read.

You can write a program to issue PCF commands to any queue manager in the network from a single node. In this way, you can both centralize and automate administration tasks.

Each PCF command is a data structure that is embedded in the application data part of a WebSphere MQ message. Each command is sent to the target queue manager using the MQI function MQPUT in the same way as any other message. The command server on the queue manager receiving the message interprets it as a command message and runs the command. To get the replies, the application issues an MQGET call and the reply data is returned in another data structure. The application can then process the reply and act accordingly.

Briefly, these are some of the things the application programmer must specify to create a PCF command message:

Message descriptor
This is a standard WebSphere MQ message descriptor, in which:

- Message type (*MsqType*) is MQMT_REQUEST.
- Message format (*Format*) is MQFMT_ADMIN.

Application data
Contains the PCF message including the PCF header, in which:

- The PCF message type (*Type*) specifies MQCFT_COMMAND.
- The command identifier specifies the command, for example, *Change Queue* (MQCMD_CHANGE_Q).

For a complete description of the PCF data structures and how to implement them, see WebSphere MQ Programmable Command Formats and Administration Interface.

Attributes in MQSC and PCF commands

Object attributes specified in MQSC commands are shown in this book in uppercase (for example, RQMNAME), although they are not case sensitive. MQSC attribute names are limited to eight characters.

Object attributes in PCF commands, which are not limited to eight characters, are shown in this book in italics. For example, the PCF equivalent of RQMNAME is *RemoteQMgrName*.

Escape PCFs

Escape PCFs are PCF commands that contain MQSC commands within the message text. You can use PCFs to send commands to a remote queue manager. For more information about using escape PCFs, see WebSphere MQ Programmable Command Formats and Administration Interface.

Using the MQAI to simplify the use of PCFs

You can use the WebSphere MQ Administration Interface (MQAI) to obtain easier programming access to PCF messages.

It performs administration tasks on a queue manager through the use of *data bags*. Data bags allow you to handle properties (or parameters) of objects in a way that is easier than using PCFs.

Use the MQAI to:

Simplify the use of PCF messages
The MQAI is an easy way to administer WebSphere MQ; you do not have to write your own PCF messages and this avoids the problems associated with complex data structures.

To pass parameters in programs that are written using MQI calls, the PCF message must contain the command and details of the string or integer data. To do this, several statements are needed in your program for every structure, and memory space must be allocated. This task is long and laborious.

On the other hand, programs written using the MQAI pass parameters into the appropriate data bag and only one statement is required for each structure. The use of MQAI data bags removes the need for you to handle arrays and allocate storage, and provides some degree of isolation from the details of the PCF.

Handle error conditions more easily
It is difficult to get return codes back from MQSC commands, but the MQAI makes it easier for the program to handle error conditions.

After you have created and populated your data bag, you can then send an administration command message to the command server of a queue manager, using the mqExecute call, which waits for any response messages. The mqExecute call handles the exchange with the command server and returns responses in a response bag.

For more information about using the MQAI and PCFs in general, see the WebSphere MQ Programmable Command Formats and Administration Interface.

Using the WebSphere MQ Explorer with WebSphere MQ for i5/OS

How to administer WebSphere MQ for i5/OS using the WebSphere MQ Explorer.

WebSphere MQ for Windows (x86 platform), and WebSphere MQ for Linux (x86 platform) provide an administration interface called the WebSphere MQ Explorer to perform administration tasks as an alternative to using CL, control or MQSC commands.

The WebSphere MQ Explorer allows you to perform local or remote administration of your network from a computer running Windows (x86 platform), or Linux (x86 platform), by pointing the WebSphere MQ Explorer at the queue managers and clusters you are interested in. The platforms and levels of WebSphere MQ that can be administered using the WebSphere MQ Explorer are described in Remote queue managers.

Using the online guidance, you can:

- Define and control various resources including queue managers, queues, channels, process definitions, client connection channels, listeners, services, namelists, and clusters.
- Start or stop a queue manager and its associated processes.
- View queue managers and their associated objects on your workstation or from other workstations.
- Check the status of queue managers, clusters, and channels.

The configuration steps you must perform on remote WebSphere MQ queue managers to allow the WebSphere MQ Explorer to administer them are outlined in "Required definitions for administration."

What you can do with the WebSphere MQ Explorer

With the WebSphere MQ Explorer, you can:

- Start and stop a queue manager (on your local machine only).
- Define, display, and alter the definitions of WebSphere MQ objects such as queues and channels.
- Browse the messages on a queue.
- Start and stop a channel.
- View status information about a channel.
- View queue managers in a cluster.
- Check to see which applications, users, or channels have a particular queue open.
- Create a new queue manager cluster using the *Create New Cluster* wizard.
- Add a queue manager to a cluster using the *Add Queue Manager to Cluster* wizard.
- Manage the authentication information object, used with Secure Sockets Layer (SSL) channel security.

Required definitions for administration

Ensure that you have satisfied the following requirements before attempting to use the WebSphere MQ Explorer to manage WebSphere MQ on a server machine. Check that:

1. A command server is running for **any** queue manager being administered, started on the server by the CL command STRMQMCSVR.
2. A suitable TCP/IP listener exists for every remote queue manager. This is the WebSphere MQ listener started by the STRMQMLSR command.
3. The server connection channel, called SYSTEM.ADMIN.SVRCONN, exists on every remote queue manager. You ***must*** create this channel yourself. It is mandatory for every remote queue manager being administered. Without it, remote administration is not possible.
4. Verify that the SYSTEM.MQEXPLORER.REPLY.MODEL queue exists.

For further information on the WebSphere MQ Explorer, see the WebSphere MQ System Administration Guide supplied with your WebSphere MQ for Windows product.

Managing the command server for remote administration

Each queue manager can have a command server associated with it. A command server processes any incoming commands from remote queue managers, or PCF commands from applications. It presents the commands to the queue manager for processing and returns a completion code or operator message depending on the origin of the command.

A command server is mandatory for all administration involving PCFs, the MQAI, and also for remote administration.

Note: For remote administration, you must ensure that the target queue manager is running. Otherwise, the messages containing commands cannot leave the queue manager from which they are issued. Instead, these messages are queued in the local transmission queue that serves the remote queue manager. Avoid this situation if at all possible.

There are separate control commands for starting and stopping the command server. You can perform the operations described in the following sections using the WebSphere MQ Explorer.

Starting the command server

To start the command server use this CL command:

```
STRMQMCSVR MQMNAME('saturn.queue.manager')
```

where `saturn.queue.manager` is the queue manager for which the command server is being started.

Displaying the status of the command server

For remote administration, ensure that the command server on the target queue manager is running. If it is not running, remote commands cannot be processed. Any messages containing commands are queued in the target queue manager's command queue (SYSTEM.ADMIN.COMMAND.QUEUE).

To display the status of the command server for a queue manager, called here `saturn.queue.manager`, the CL command is:

```
DSPMQMCSVR MQMNAME('saturn.queue.manager')
```

Issue this command on the target machine. If the command server is running, the panel shown in Figure 6 appears:

```
               Display MQM Command Server (DSPMQMCSVR)


Queue manager name . . . . . . . > saturn.queue.manager

MQM Command Server Status. . . . > RUNNING









F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 6. Display MQM Command Server panel

Stopping a command server

To end a command server, the command, using the previous example is:

```
ENDMQMCSVR MQMNAME('saturn.queue.manager')
```

You can stop the command server in two different ways:

- For a controlled stop, use the ENDMQMCSVR command with the *CNTRLD option, which is the default.

- For an immediate stop, use the ENDMQMCSVR command with the *IMMED option.

Note: Stopping a queue manager also ends the command server associated with it (if one has been started).

Instrumentation events

You can use WebSphere MQ instrumentation events to monitor the operation of queue managers. See Monitoring WebSphere MQ for information about WebSphere MQ instrumentation events and how to use them.

Chapter 4. Work management

This chapter describes the way in which WebSphere MQ handles work requests, and details the options available for prioritizing and controlling the jobs associated with WebSphere MQ.

Warning

Do **not** alter WebSphere MQ work management objects unless you fully understand the concepts of i5/OS and WebSphere MQ work management. Additional information regarding subsystems and job descriptions can be found under *Work Management* in the i5/OS Information Center, see http://publib.boulder.ibm.com/iseries/. Pay particular attention to the sections on "Job Starting and Routing" and "Batch Jobs".

WebSphere MQ for i5/OS incorporates the i5/OS UNIX® environment and i5/OS threads. Do **not** make any changes to the objects in the Integrated File System (IFS).

During normal operations, a WebSphere MQ queue manager starts a number of batch jobs to perform different tasks. By default these batch jobs run in the QMQM subsystem that is created when WebSphere MQ is installed.

Work management refers to the process of tailoring WebSphere MQ tasks to obtain the optimum performance from your system, or to make administration simpler.

For example, you can:

- Change the run-priority of jobs to make one queue manager more responsive than another.
- Redirect the output of a number of jobs to a particular output queue.
- Make all jobs of a certain type run in a specific subsystem.
- Isolate errors to a subsystem.

Work management is carried out by creating or changing the job descriptions associated with the WebSphere MQ jobs. You can configure work management for:

- An entire WebSphere MQ installation
- Individual queue managers
- Individual jobs for individual queue managers

Description of WebSphere MQ tasks

This is a table of the WebSphere MQ jobs and a brief description of each.

When a queue manager is running, you see some or all of the following batch jobs running under the QMQM user profile in the WebSphere MQ subsystem. The jobs are described briefly in Table 1 on page 44.

You can view all jobs connected to a queue manager using option 22 on the Work with Queue Manager (WRKMQM) panel. You can view listeners using the WRKMQMLSR command.

Table 1. WebSphere MQ tasks.

Job name	Function
AMQALMPX	The checkpoint processor that periodically takes journal checkpoints.
AMQZMUC0	Utility manager. This job executes critical queue manager utilities, for example the journal chain manager.
AMQZXMA0	The execution controller that is the first job started by the queue manager. It handles MQCONN requests, and starts agent processes to process WebSphere MQ API calls.
AMQZFUMA	Object authority manager (OAM).
AMQZLAA0	Queue manager agents that perform most of the work for applications that connect to the queue manager using MQCNO_STANDARD_BINDING.
AMQZLAS0	Queue manager agent.
AMQZMUF0	Utility Manager
AMQZMGR0	Process controller. This job is used to start up and manage listeners and services.
AMQZMUR0	Utility manager. This job executes critical queue manager utilities, for example the journal chain manager.
AMQFQPUB	Queued publish/subscribe daemon.
AMQFCXBA	Broker worker job.
RUNMQBRK	Broker control job.
AMQRMPPA	Channel process pooling job.
AMQCRSTA	TCP/IP-invoked channel responder.
AMQCRS6B	LU62 receiver channel and client connection (see note).
AMQRRMFA	Repository manager for clusters.
AMQCLMAA	Non-threaded TCP/IP listener.
AMQPCSEA	PCF command processor that handles PCF and remote administration requests.
RUNMQTRM	Trigger monitor.
RUNMQDLQ	Dead letter queue handler.
RUNMQCHI	The channel initiator.
RUNMQCHL	Sender channel job that is started for each sender channel.
RUNMQLSR	Threaded TCP/IP listener.
AMQXSSVN	Shared memory servers.
AMQRCMLA	Channel MQSC and PCF command processor.
AMQZTRCN	Trace.
Note: The LU62 receiver job runs in the communications subsystem and takes its run-time properties from the routing and communications entries that are used to start the job. See WebSphere MQ Intercommunication for more details.	

WebSphere MQ work management objects

When WebSphere MQ is installed, various objects are supplied in the QMQM library to assist with work management. These objects are the ones necessary for WebSphere MQ jobs to run in their own subsystem.

Sample job descriptions are provided for two of the WebSphere MQ batch jobs. If no specific job description is provided for a WebSphere MQ job, it runs with the default job description QMQMJOBD.

The work management objects that are supplied when you install WebSphere MQ are listed in Table 2 and the objects created for a queue manager are listed in Table 3

Note: The work management objects can be found in the QMQM library and the queue manager objects can be found in the queue manager library.

Table 2. Work management objects

Name	Type	Description
AMQALMPX	*JOBD	The job description that is used by the checkpoint process
AMQZLAA0	*JOBD	The job description that is used by the WebSphere MQ agent processes
AMQZLSA0	*JOBD	The isolated bindings queue manager agent
AMQZXMA0	*JOBD	The job description that is used by WebSphere MQ execution controllers
QMQM	*SBSD	The subsystem in which all WebSphere MQ jobs run
QMQM	*JOBQ	The job queue attached to the supplied subsystem
QMQMJOBD	*JOBD	The default WebSphere MQ job description, used if there is not a specific job description for a job
QMQMMSG	*MSGQ	The default message queue for WebSphere MQ jobs.
QMQMRUN20	*CLS	A class description for high priority WebSphere MQ jobs
QMQMRUN35	*CLS	A class description for medium priority WebSphere MQ jobs
QMQMRUN50	*CLS	A class description for low priority WebSphere MQ jobs

Table 3. Work management objects created for a queue manager

Name	Type	Description
AMQA000000	*JRNRCV	Local journal receiver
AMQAJRN	*JRN	Local journal
AMQJRNINF	*USRSPC	User space that is updated with the latest journal receivers required for startup and media recovery of a queue manager. This user space can be queried by an application to determine which journal receivers require archiving and which can be safely deleted.
AMQAJRNMSG	*MSGQ	Local journal message queue
AMQCRC6B	*PGM	Program to start the LU6.2 connection
AMQRFOLD	*FILE	Migrated queue manager channel definition file
QMQMMSG	*MSGQ	Queue manager message queue

How WebSphere MQ uses the work management objects

Warning

Do not alter the job queue entry settings in the QMQM subsytem to limit the number of jobs allowed in the subsystem by priority. If you attempt to do this, you can stop essential WebSphere MQ jobs from running after they are submitted and cause the queue manager startup to fail.

To understand how to configure work management, you must first understand how WebSphere MQ uses job descriptions.

The job description used to start the job controls many attributes of the job. For example:

- The job queue on which the job is queued and on which subsystem the job runs.
- The routing data used to start the job and class that the job uses for its run-time parameters.
- The output queue that the job uses for print files.

The process of starting a WebSphere MQ job can be considered in three steps:

1. WebSphere MQ selects a job description.

   WebSphere MQ uses the following technique to determine which job description to use for a batch job:

   a. Look in the queue manager library for a job description with the same name as the job. See "Understanding WebSphere MQ queue manager library names" on page 681 for further details about the queue manager library.
   b. Look in the queue manager library for the default job description QMQMJOBD.
   c. Look in the QMQM library for a job description with the same name as the job.
   d. Use the default job description, QMQMJOBD, in the QMQM library.

2. The job is submitted to the job queue.

   Job descriptions supplied with WebSphere MQ have been set up, by default, to put jobs on to job queue QMQM in library QMQM. The QMQM job queue is attached to the supplied QMQM subsystem, so by default the jobs start running in the QMQM subsystem.

3. The job enters the subsystem and goes through the routing steps.

   When the job enters the subsystem, the routing data specified on the job description is used to find routing entries for the job.

   The routing data must match one of the routing entries defined in the QMQM subsystem, and this defines which of the supplied classes (QMQMRUN20, QMQMRUN35, or QMQMRUN50) is used by the job.

Note: If WebSphere MQ jobs do not appear to be starting, make sure that the subsystem is running and the job queue is not held,

If you have modified the WebSphere MQ work management objects, make sure everything is associated correctly. For example, if you specify a job queue other than QMQM/QMQM on the job description, make sure that an ADDJOBQE is performed for the subsystem, that is, QMQM.

You can create a job description for each job documented in Table 1 on page 44 using the following worksheet as an example:

```
What is the queue manager library name? ___________
Does job description AMQZXMA0 exist in the queue manager library? Yes      No
Does job description QMQMJOBD exist in the queue manager library? Yes      No
Does job description AMQZXMA0 exist in the QMQM library?          Yes      No
Does job description QMQMJOBD exist in the QMQM library?          Yes      No
```

If you answer No to all these questions, create a global job description QMQMJOBD in the QMQM library.

The WebSphere MQ message queue

A WebSphere MQ message queue, QMQMMSG, is created in each queue manager library. Operating system messages are sent to this queue when queue manager jobs end and WebSphere MQ sends messages to the queue. For example, to report which journal receivers are needed at startup. Keep the number of messages in this message queue at a manageable size to make it easier to monitor.

Default system examples

The following examples show how an unmodified WebSphere MQ installation works when some of the standard jobs are submitted at queue manager startup time.

The first job that is started is the execution controller, AMQZXMA0.

1. Issue the **STRMQM** command for queue manager TESTQM.
2. WebSphere MQ searches the queue manager library QMTESTQM, firstly for job description AMQZXMA0, and then job description QMQMJOBD.

   Neither of these job descriptions exist, so WebSphere MQ looks for job description AMQZXMA0 in the product library QMQM. This job description exists, so it is used to submit the job.
3. The job description uses the WebSphere MQ default job queue, so the job is submitted to job queue QMQM/QMQM.
4. The routing data on the AMQZXMA0 job description is QMQMRUN20, so the system searches the subsystem routing entries for one that matches that data.

   By default, the routing entry with sequence number 9900 has comparison data that matches QMQMRUN20, so the job is started with the class defined on that routing entry, which is also called QMQMRUN20.
5. The QMQM/QMQMRUN20 class has run priority set to 20, so the AMQZXMA0 job runs in subsystem QMQM with the same priority as most interactive jobs on the system.

The next job that starts is the checkpoint process, AMQALMPX.

1. WebSphere MQ searches the queue manager library QMTESTQM, firstly for job description AMQALPMX, and then job description QMQMJOBD.

   Neither of these job descriptions exist, so WebSphere MQ looks for job descriptions AMQALMPX and QMQMJOBD in the product library QMQM.

   Job description AMQALMPX does not exist but QMQMJOBD does, so QMQMJOBD is used to submit the job.

   Note: The QMQMJOBD job description is always used for WebSphere MQ jobs that do not have their own job description.
2. The job description uses the WebSphere MQ default job queue, so the job is submitted to job queue QMQM/QMQM.
3. The routing data on the QMQMJOBD job description is QMQMRUN35, so the system searches the subsystem routing entries for one that matches that data.

   By default, the routing entry with sequence number 9910 has comparison data that matches QMQMRUN35, so the job is started with the class defined on that routing entry, which is also called QMQMRUN35.
4. The QMQM/QMQMRUN35 class has run priority set to 35, so the AMQALMPX job runs in subsystem QMQM with a lower priority than most interactive jobs on the system, but higher priority than most batch jobs.

Configuring work management

The preceding examples show how WebSphere MQ job descriptions determine the run-time attributes of WebSphere MQ jobs.

The following examples show how you can change and create WebSphere MQ job descriptions to change the run-time attributes of WebSphere MQ jobs.

The key to the flexibility of WebSphere MQ work management lies in the two-tier way that WebSphere MQ searches for job descriptions:

- If you create or change job descriptions in a queue manager library, those changes override the global job descriptions in QMQM, but the changes are *local* and affect that particular queue manager alone.
- If you create or change global job descriptions in the QMQM library, those job descriptions affect all queue managers on the system, unless overridden locally for individual queue managers.

Configuration examples

1. The following example increases the priority of channel control jobs for an individual queue manager.

   To make the repository manager and channel initiator jobs, AMQRRMFA and RUNMQCHI, run as quickly as possible for queue manager TESTQM, carry out the following steps:

   a. Create local duplicates of the QMQM/QMQMJOBD job description with the names of the WebSphere MQ processes that you want to control in the queue manager library. For example,

   ```
    CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
   NEWOBJ(RUNMQCHI)
    CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
   NEWOBJ(AMQRRMFA)
   ```

   b. Change the routing data parameter on the job description to ensure that the jobs use the QMQMRUN20 class.

   ```
    CHGJOBD JOBD(QMTESTQM/RUNMQCHI) RTGDTA('QMQMRUN20')
    CHGJOBD JOBD(QMTESTQM/AMQRRMFA) RTGDTA('QMQMRUN20')
   ```

   The AMQRRMFA and RUNMQCHI jobs for queue manager TESTQM now:

   - Use the new local job descriptions in the queue manager library
   - Run with priority 20, because the QMQMRUN20 class is used when the jobs enter the subsystem.

2. The following example defines a new run priority class for the QMQM subsystem.

   a. Create a duplicate class in the QMQM library, to allow other queue managers to access the class, by issuing the following command:

   ```
   CRTDUPOBJ OBJ(QMQMRUN20) FROMLIB(QMQM) OBJTYPE(*CLS) TOLIB(QMQM)
          NEWOBJ(QMQMRUN10)
   ```

   b. Change the class to have the new run priority by issuing the following command:

   ```
   CHGCLS CLS(QMQM/QMQMRUN10) RUNPTY(10)
   ```

   c. Add the new class definition to the subsystem by issuing the following command:

   ```
     ADDRTGE SBSD(QMQM/QMQM) SEQNBR(8999) CMPVAL('QMQMRUN10') PGM(QSYS/QCMD)
             CLS(QMQM/QMQMRUN10)
   ```

Note: You can specify any numeric value for the routing sequence number, but the values must be in sequential order. This sequence number tells the subsystem the order in which routing entries are to be searched for a routing data match.

d. Change the local or global job description to use the new priority class by issuing the following command:

```
CHGJOBD JOBD(QMQMlibname/QMQMJOBD) RTGDTA('QMQMRUN10')
```

Now all the queue manager jobs associated with the QMlibraryname use a run priority of 10.

3. The following example runs a queue manager in its own subsystem

To make all the jobs for queue manager TESTQM run in the QBATCH subsystem, carry out the following steps:

a. Create a local duplicate of the QMQM/QMQMJOBD job description in the queue manager library with the command

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
```

b. Change the job queue parameter on the job description to ensure that the jobs use the QBATCH job queue.

```
CHGJOBD JOBD(QMTESTQM/QMQMJOBD) JOBQ(*LIBL/QBATCH)
```

Note: The job queue is associated with the subsystem description. If you find that the jobs are staying on the job queue, verify that the job queue definition is defined on the SBSD. Use the DSPSBSD command for the subsystem and take option **6**, "Job queue entries".

All jobs for queue manager TESTQM now:

- Use the new local default job description in the queue manager library
- Are submitted to job queue QBATCH.

To ensure that jobs are routed and prioritized correctly:

- Either create routing entries for the WebSphere MQ jobs in subsystem QBATCH, or
- Rely on a catch-all routing entry that calls QCMD, irrespective of what routing data is used.

  This option works only if the maximum active jobs option for job queue QBATCH is set to *NOMAX. The system default is 1.

4. The following example creates another WebSphere MQ subsystem

a. Create a duplicate subsystem in the QMQM library by issuing the following command:

```
CRTDUPOBJ OBJ(QMQM) FROMLIB(QMQM) OBJTYPE(*SBSD) TOLIB(QMQM) NEWOBJ(QMQM2)
```

b. Remove the QMQM job queue by issuing the following command:

```
RMVJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM)
```

c. Create a new job queue for the subsystem by issuing the following command:

```
CRTJOBQ JOBQ(QMQM/QMQM2) TEXT('Job queue for MQSeries Queue Manager')
```

d. Add a job queue entry to the subsystem by issuing the following command:

```
ADDJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM2) MAXACT(*NOMAX)
```

e. Create a duplicate QMQMJOBD in the queue manager library by issuing the following command:

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMlibraryname)
```

f. Change the job description to use the new job queue by issuing the following command:

```
CHGJOBD JOBD(QMlibraryname/QMQMJOBD) JOBQ(QMQM/QMQM2)
```

g. Start the subsystem by issuing the following command:

```
STRSBS SBSD(QMQM/QMQM2)
```

Note:

a. You can specify the subsystem in any library. If for any reason the product is reinstalled, or the QMQM library is replaced, any changes you made are removed.

b. All the queue manager jobs associated with the QMlibraryname now run under subsystem QMQM2.

5. The following example collects all output for a job type.

   To collect all the checkpoint process, AMQALMPX, job logs for multiple queue managers onto a single output queue, carry out the following steps:

   a. Create an output queue, for example

   ```
   CRTOUTQ OUTQ(MYLIB/CHCKPTLOGS)
   ```

   b. Create a global duplicate of the QMQM/QMQMJOBD job description, using the name of the WebSphere MQ process that you want to control, for example

   ```
   CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) NEWOBJ(AMQALMPX)
   ```

   c. Change the output queue parameter on the job description to point to your new output queue, and change the job logging level so that all messages are written to the job log.

   ```
   CHGJOBD JOBD(QMQM/AMQALMPX) OUTQ(MYLIB/CHKPTLOGS) LOG(4 00 *SECLVL)
   ```

   All WebSphere MQ AMQALMPX jobs, for all queue managers, use the new global AMQALMPX job description, providing that there are no local overriding job descriptions in the local queue manager library.

   All job log spool files for these jobs are now written to output queue CHKPTLOGS in library MYLIB.

   Note:

   a. The preceding example works only if the QPJOBLOG, or any print file, has a value of *JOB for its output queue parameter. In the preceding example, the QSYS/QPDJOBLOG file needs OUTQ set to *JOB.

   b. To change a system print file, use the CHGPRTF command. For example:

   ```
   CHGPRTF PRTF(QJOBLOG) OUTQ(*JOB)
   ```

   The *JOB option indicates that your job descriptions must be used.

   c. You can send any spool files associated with the WebSphere MQ jobs to a particular output queue. However, verify that the print file being used has the appropriate value for the OUTQ parameter.

Chapter 5. Protecting WebSphere MQ objects

Security for WebSphere MQ for i5/OS is implemented using the WebSphere MQ Object Authority Manager (OAM) in conjunction with i5/OS object level security.

Security considerations

Security considerations that must be made when determining access authority to WebSphere MQ objects.

You need to consider the following points when setting up authorities to the users in your enterprise:

1. Grant and revoke authorities to the WebSphere MQ for i5/OS commands using the i5/OS **GRTOBJAUT** and **RVKOBJAUT** commands.

   In the QMQM library certain non command (*cmd) objects are set to have ***PUBLIC** authority to ***USE**. Do not change the authorities of these objects or use an authorization list to provide authority. Any incorrect authority might break WebSphere MQ functionality.
2. During installation of WebSphere MQ for i5/OS the following special user profiles are created:

   QMQM
   Is used primarily for internal product-only functions. However, it can be used to run trusted applications using MQCNO_FASTPATH_BINDINGS; see the WebSphere MQ Application Programming Guide for further information.

   QMQMADM
   Is used as a group profile for administrators of WebSphere MQ. The group profile gives access to CL commands and WebSphere MQ resources.
3. If you are sending channel commands to remote queue managers, ensure that your user profile is a member of the group QMQMADM on the target system. For a list of PCF and MQSC channel commands, see “Channel command security” on page 81.
4. The group set associated with a user is cached when the group authorizations are computed by the OAM.

   Any changes made to a user’s group memberships after the group set has been cached are not recognized until you restart the queue manager or execute RFRMQMAUT to refresh security.
5. Limit the number of users who have authority to work with commands that are particularly sensitive. These commands include:
   - Create Message Queue Manager (**CRTMQM**)
   - Delete Message Queue Manager (**DLTMQM**)
   - Start Message Queue Manager (**STRMQM**)
   - End Message Queue Manager (**ENDMQM**)
   - Start Command Server (**STRMQMCSVR**)
   - End Command Server (**ENDMQMCSVR**)

6. Channel definitions contain a security exit program specification. Channel creation and modification requires special considerations. Details of security exits is given in WebSphere MQ Intercommunication.
7. The channel exit and trigger monitor programs can be substituted. The security of such replacements is the responsibility of the programmer.

Understanding the Object Authority Manager

The OAM manages users' authorizations to manipulate WebSphere MQ objects, including queues and process definitions. It also provides a command interface through which you can grant or revoke access authority to an object for a specific group of users. The decision to allow access to a resource is made by the OAM, and the queue manager follows that decision. If the OAM cannot make a decision, the queue manager prevents access to that resource.

Resources you can protect with the OAM

Through the OAM you can control:

- Access to WebSphere MQ objects through the MQI. When an application program attempts to access an object, the OAM checks that the user profile making the request has the authorization for the operation requested.

  In particular, this means that queues, and the messages on queues, can be protected from unauthorized access.
- Permission to use PCF and MQSC commands.

Different groups of users can have different kinds of access authority to the same object. For example, for a specific queue, one group could perform both put and get operations; another group might be allowed only to browse the queue (MQGET with browse option). Similarly, some groups might have get and put authority to a queue, but not be allowed to alter or delete the queue.

WebSphere MQ for i5/OS provides commands to grant, revoke, and display the authority that an application or user has to do the following:

- Issue WebSphere MQ for i5/OS commands
- Perform operations on WebSphere MQ for i5/OS objects

WebSphere MQ authorities

Access to WebSphere MQ objects is controlled by authorities to:

1. Issue the WebSphere MQ command
2. Access the WebSphere MQ objects referenced by the command

All WebSphere MQ for i5/OS CL commands are shipped with an owner of QMQM, and the administration profile (QMQMADM) has *USE rights with the *PUBLIC access set to *EXCLUDE.

Changes to the authority structure of some of the product's CL commands allows public use of these commands, provided that you have the required OAM authority to the WebSphere MQ objects to make these changes.

Access authorities for WebSphere MQ objects

Access authorities required for running WebSphere MQ CL commands.

WebSphere MQ for i5/OS categorizes the product's CL commands into two groups:

Group 1

Users must be in the QMQMADM user group, or have *ALLOBJ authority, to process these commands. Users having either of these authorities can process all commands in all categories without requiring any extra authority.

Note: These authorities override any OAM authority.

These commands can be grouped as follows:

- Command Server Commands
  - ENDMQMCSVR, End WebSphere MQ Command Server
  - STRMQMCSVR, Start WebSphere MQ Command Server
- Dead-Letter Queue Handler Command
  - STRMQMDLQ, Start WebSphere MQ Dead-Letter Queue Handler
- Listener Command
  - ENDMQMLSR, End WebSphere MQ listener
  - STRMQMLSR, Start non-object listener
- Media Recovery Commands
  - RCDMQMIMG, Record WebSphere MQ Object Image
  - RCRMQMOBJ, Recreate WebSphere MQ Object
  - WRKMQMTRN, Work with WebSphere MQ Transactions
- Queue Manager Commands
  - CRTMQM, Create Message Queue Manager
  - DLTMQM, Delete Message Queue Manager
  - ENDMQM, End Message Queue Manager
  - STRMQM, Start Message Queue Manager
- Security Commands
  - GRTMQMAUT, Grant WebSphere MQ Object Authority
  - RVKMQMAUT, Revoke WebSphere MQ Object Authority
- Trace Command
  - TRCMQM, Trace WebSphere MQ Job
- Transaction Commands
  - RSVMQMTRN, Resolve WebSphere MQ Transaction
- Trigger Monitor Commands
  - STRMQMTRM, Start Trigger Monitor
- WebSphere MQSC Commands
  - RUNMQSC, Run WebSphere MQSC Commands
  - STRMQMMQSC, Start WebSphere MQSC Commands

Group 2

The rest of the commands, for which two levels of authority are required:

1. i5/OS authority to run the command. A WebSphere MQ administrator sets this using the **GRTOBJAUT** command to override the *PUBLIC(*EXCLUDE) restriction for a user or group of users.

   For example:

   ```
   GRTOBJAUT OBJ(DSPMQMQ) OBJTYPE(*CMD) USER(MQUSER) AUT(*USE)
   ```

2. WebSphere MQ authority to manipulate the WebSphere MQ objects associated with the command, or commands, given the correct i5/OS authority in Step 1.

   This authority is controlled by the user having the appropriate OAM authority for the required action, set by a WebSphere MQ administrator using the **GRTMQMAUT** command

   For example:

   ```
   CHGMQMQ *connect authority to the queue manager + *admchg authority to
     the queue
   ```

The commands can be grouped as follows:

- Authentication Information Commands
  - CHGMQMAUTI, Change WebSphere MQ Authentication Information
  - CPYMQMAUTI, Copy WebSphere MQ Authentication Information
  - CRTMQMAUTI, Create WebSphere MQ Authentication Information
  - DLTMQMAUTI, Delete WebSphere MQ Authentication Information
- Channel Commands
  - CHGMQMCHL, Change WebSphere MQ Channel
  - CPYMQMCHL, Copy WebSphere MQ Channel
  - CRTMQMCHL, Create WebSphere MQ Channel
  - DLTMQMCHL, Delete WebSphere MQ Channel
  - RSVMQMCHL, Resolve WebSphere MQ Channel
- Display commands

  To process the DSP commands you must grant the user `*connect` and `*admdsp` authority to the queue manager, together with any specific option listed:
  - DSPMQM, Display Message Queue Manager
  - DSPMQMAUT, Display WebSphere MQ Object Authority
  - DSPMQMAUTI, Display WebSphere MQ Authentication Information
  - DSPMQMCHL, Display WebSphere MQ Channel
  - DSPMQMCSVR, Display WebSphere MQ Command Server
  - DSPMQMNL, Display WebSphere MQ Namelist – `*admdsp` to the namelist
  - DSPMQMOBJN, Display WebSphere MQ Object Names
  - DSPMQMPRC, Display WebSphere MQ Process – `*admdsp` to the process
  - DSPMQMQ, Display WebSphere MQ Queue – `*admdsp` to the queue
  - DSPMQMTOP, Display WebSphere MQ Topic – `*admdsp` to the topic
- Work with commands

  To process the WRK commands and display the options panel you must grant the user `*connect` and `*admdsp` authority to the queue manager, together with any specific option listed:
  - WRKMQM, Work with Message Queue Managers
  - WRKMQMAUT, Work with WebSphere MQ Object Authority

- WRKMQMAUTD, Work with WebSphere MQ Object Authority Data
- WRKMQMAUTI, Work with WebSphere MQ Authentication Information
- WRKMQMCHL, Work with WebSphere MQ Channel
- WRKMQMCHST, Work with WebSphere MQ Channel Status
- WRKMQMCL, Work with WebSphere MQ Clusters
- WRKMQMCLQ, Work with WebSphere MQ Cluster Queues
- WRKMQMCLQM, Work with WebSphere MQ Cluster Queue Manager
- WRKMQMLSR, Work with WebSphere MQ Listener
- WRKMQMMSG, Work with WebSphere MQ Messages

  This requires `*browse` authority to the queue
- WRKMQMNL, Work with WebSphere MQ Namelists

  This requires the following authorities:
  - `*admchg` for the Change WebSphere MQ Namelist command.
  - `*admcpy` for the Copy WebSphere MQ Namelist command.
  - `*admcrt` for the Create WebSphere MQ Namelist command.
  - `*admdlt` for the Delete WebSphere MQ Namelist command.
  - `*admdsp` for the Display WebSphere MQ Namelist command.
- WRKMQMPRC, Work with WebSphere MQ Processes

  This requires the following authorities:
  - `*admchg` for the Change WebSphere MQ Process command.
  - `*admcpy` for the Copy WebSphere MQ Process command.
  - `*admcrt` for the Create WebSphere MQ Process command.
  - `*admdlt` for the Delete WebSphere MQ Process command.
  - `*admdsp` for the Display WebSphere MQ Process command.
- WRKMQMQ, Work with WebSphere MQ queues

  This requires the following authorities:
  - `*admchg` for the Change WebSphere MQ Queue command.
  - `*admclr` for the Clear WebSphere MQ Queue command.
  - `*admcpy` for the Copy WebSphere MQ Queue command.
  - `*admcrt` for the Create WebSphere MQ Queue command.
  - `*admdlt` for the Delete WebSphere MQ Queue command.
  - `*admdsp` for the Display WebSphere MQ Queue command.
- WRKMQMQSTS, Work with WebSphere MQ Queue Status
- WRKMQMTOP, Work with WebSphere MQ Topics

• Other Channel commands

  To process the channel commands you must grant the user the specific authorities listed:
- ENDMQMCHL, End WebSphere MQ Channel

  This requires `*connect` authority to the queue manager and `*allmqi` authority to the transmission queue associated with the channel.
- ENDMQMLSR, End WebSphere MQ Listener

  This requires `*connect` authority to the queue manager and `*ctrl` authority to the named listener object.
- PNGMQMCHL, Ping WebSphere MQ Channel

This requires *connect* and *inq*authority to the queue manager.

- RSTMQMCHL, Reset WebSphere MQ Channel

  This requires *connect* authority to the queue manager.

- STRMQMCHL, Start WebSphere MQ Channel

  This requires *connect* authority to the queue manager and *ctrl* authority to the channel object.

- STRMQMCHLI, Start WebSphere MQ Channel Initiator

  This requires *connect* and *inq* authority to the queue manager, and *allmqi* authority to the initiation queue associated with the transmission queue of the channel.

- STRMQMLSR, Start WebSphere MQ Listener

  This requires *connect authority to the queue manager and *ctrl authority to the named listener object.

- Other commands:

  To process the following commands you must grant the user the specific authorities listed:

  - CCTMQM, Connect to Message Queue Manager

    This requires no WebSphere MQ object authority.

  - CHGMQM, Change Message Queue Manager

    This requires *connect* and *admchg* authority to the queue manager.

  - CHGMQMNL, Change WebSphere MQ Namelist

    This requires *connect* authority to the queue manager and *admchg* authority to the namelist.

  - CHGMQMPRC, Change WebSphere MQ Process

    This requires *connect* authority to the queue manager and *admchg* authority to the process.

  - CHGMQMQ, Change WebSphere MQ Queue

    This requires *connect* authority to the queue manager and *admchg* authority to the queue.

  - CLRMQMQ, Clear WebSphere MQ Queue

    This requires *connect* authority to the queue manager and *admchg* authority to the queue.

  - CPYMQMNL, Copy WebSphere MQ Namelist

    This requires *connect* and *admcrtauthority to the queue manager.

  - CPYMQMPRC, Copy WebSphere MQ Process

    This requires *connect* and *admcrtauthority to the queue manager.

  - CPYMQMQ, Copy WebSphere MQ Queue

    This requires *connect* and *admcrtauthority to the queue manager.

  - CRTMQMNL, Create WebSphere MQ Namelist

    This requires *connect* and *admcrtauthority to the queue manager and *admdsp* authority to the default namelist.

  - CRTMQMPRC, Create WebSphere MQ Process

    This requires *connect* and *admcrtauthority to the queue manager and *admdsp* authority to the default process.

  - CRTMQMQ, Create WebSphere MQ Queue

    This requires *connect* and *admcrtauthority to the queue manager and *admdsp* authority to the default queue.

- CVTMQMDTA, Convert WebSphere MQ Data Type Command

  This requires no WebSphere MQ object authority.
- DLTMQMNL, Delete WebSphere MQ Namelist

  This requires `*connect` authority to the queue manager and `*admdlt` authority to the namelist.
- DLTMQMPRC, Delete WebSphere MQ Process

  This requires `*connect` authority to the queue manager and `*admdlt` authority to the process.
- DLTMQMQ, Delete WebSphere MQ Queue

  This requires `*connect` authority to the queue manager and `*admdlt` authority to the queue.
- DSCMQM, Disconnect from Message Queue Manager

  This requires no WebSphere MQ object authority.
- RFRMQMAUT, Refresh Security

  This requires `*connect` authority to the queue manager.
- RFRMQMCL, Refresh Cluster

  This requires `*connect` authority to the queue manager.
- RSMMQMCLQM, Resume Cluster Queue Manager

  This requires `*connect` authority to the queue manager.
- RSTMQMCL, Reset Cluster

  This requires `*connect` authority to the queue manager.
- SPDMQMCLQM, Suspend Cluster Queue Manager

  This requires `*connect` authority to the queue manager.

Access authorizations

Authorizations defined by the AUT keyword on the **GRTMQMAUT** and **RVKMQMAUT** commands can be categorized as follows:

- Authorizations related to MQI calls
- Authorization-related administration commands
- Context authorizations
- General authorizations, that is, for MQI calls, for commands, or both

The following tables list the different authorities, using the AUT parameter for MQI calls, Context calls, MQSC and PCF commands, and generic operations.

Table 4. Authorizations for MQI calls

AUT	Description
***ALTUSR**	Allow another user's authority to be used for MQOPEN and MQPUT1 calls.
***BROWSE**	Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.
***CONNECT**	Connect the application to the specified queue manager by issuing an MQCONN call.
***GET**	Retrieve a message from a queue by issuing an MQGET call.
***INQ**	Make an inquiry on a specific queue by issuing an MQINQ call.
***PUT**	Put a message on a specific queue by issuing an MQPUT call.

Table 4. Authorizations for MQI calls (continued)

AUT	Description
***SET**	Set attributes on a queue from the MQI by issuing an MQSET call. If you open a queue for multiple options, you must be authorized for each of them.

Table 5. Authorizations for context calls

AUT	Description
***PASSALL**	Pass all context on the specified queue. All the context fields are copied from the original request.
***PASSID**	Pass identity context on the specified queue. The identity context is the same as that of the request.
***SETALL**	Set all context on the specified queue. This is used by special system utilities.
***SETID**	Set identity context on the specified queue. This is used by special system utilities.

Table 6. Authorizations for MQSC and PCF calls

AUT	Description
***ADMCHG**	Change the attributes of the specified object.
***ADMCLR**	Clear the specified queue (PCF Clear queue command only).
***ADMCRT**	Create objects of the specified type.
***ADMDLT**	Delete the specified object.
***ADMDSP**	Display the attributes of the specified object.

Table 7. Authorizations for generic operations

AUT	Description
***ALL**	Use all operations applicable to the object.
***ALLADM**	Perform all administration operations applicable to the object.
***ALLMQI**	Use all MQI calls applicable to the object.
***CTRL**	Control startup and shutdown of channels, listeners, and services.
***CTRLX**	Reset sequence number and resolve indoubt channels.

Using the GRTMQMAUT command

Provided that you have the required authorization, you can use the **GRTMQMAUT** command to grant authorization of a user profile or user group to access a particular object. The following examples illustrate how the **GRTMQMAUT** command is used:

1.
```
GRTMQMAUT OBJ(RED.LOCAL.QUEUE) OBJTYPE(*LCLQ) USER(GROUPA) +
          AUT(*BROWSE *PUT) MQMNAME('saturn.queue.manager')
```

   In this example:

   - `RED.LOCAL.QUEUE` is the object name.
   - `*LCLQ` (local queue) is the object type.
   - `GROUPA` is the name of a user profile on the system whose authorizations are to change. This can be used as a group profile for other users.
   - `*BROWSE` and `*PUT` are the authorizations being granted to the specified queue.

*BROWSE adds authorization to browse messages on the queue (to issue MQGET with the browse option).

*PUT adds authorization to put (MQPUT) messages on the queue.

- saturn.queue.manager is the queue manager name.

2. The following command grants to users JACK and JILL all applicable authorizations, to all process definitions, for the default queue manager.

```
GRTMQMAUT OBJ(*ALL) OBJTYPE(*PRC) USER(JACK JILL) AUT(*ALL)
```

3. The following command grants user GEORGE authority to put a message on the queue ORDERS, on the queue manager TRENT.

```
GRTMQMAUT OBJ(TRENT) OBJTYPE(*MQM) USER(GEORGE) AUT(*CONNECT) MQMNAME (TRENT)
GRTMQMAUT OBJ(ORDERS) OBJTYPE(*Q) USER(GEORGE) AUT(*PUT) MQMNAME (TRENT)
```

Using the RVKMQMAUT command

Provided that you have the required authorization, you can use the **RVKMQMAUT** command to remove previously granted authorization of a user profile or user group to access a particular object. The following examples illustrate how the **RVKMQMAUT** command is used:

1.
```
RVKMQMAUT OBJ(RED.LOCAL.QUEUE) OBJTYPE(*LCLQ) USER(GROUPA) +
AUT(*PUT) MQMNAME('saturn.queue.manager')
```

The authority to put messages to the specified queue, that was granted in the previous example, is removed for GROUPA.

2.
```
RVKMQMAUT OBJ(PAY*) OBJTYPE(*Q) USER(*PUBLIC) AUT(*GET) +
MQMNAME(PAYROLLQM)
```

Authority to get messages from any queue whose name starts with the characters PAY, owned by queue manager PAYROLLQM, is removed from all users of the system unless they, or a group to which they belong, have been separately authorized.

Using the DSPMQMAUT command

The display MQM authority (**DSPMQMAUT**) command shows, for the specified object and user, the list of authorizations that the user has for the object. The following example illustrates how the command is used:

```
DSPMQMAUT OBJ(ADMINNL) OBJTYPE(*NMLIST) USER(JOE) OUTPUT(*PRINT) +
MQMNAME(ADMINQM)
```

Using the RFRMQMAUT command

The refresh MQM security (**RFRMQMAUT**) command enables you to update the OAM's authorization group information immediately, reflecting changes made at the operating system level, without needing to stop and restart the queue manager. The following example illustrates how the command is used:

```
RFRMQMAUT MQMNAME(ADMINQM)
```

Understanding the authorization specification tables

The authorization specification tables starting in topic Table 8 on page 60 define precisely how the authorizations work and the restrictions that apply. The tables apply to these situations:

- Applications that issue MQI calls
- Administration programs that issue MQSC commands as escape PCFs

- Administration programs that issue PCF commands

In this section the information is presented as a set of tables that specify the following:

Action to be performed
MQI option, MQSC command, or PCF command.

Access control object
Queue, process definition, queue manager, namelist, channel, client connection channel, listener, service, or authentication information object.

Authorization required
Expressed as an MQZAO_ constant.

In the tables, the constants prefixed by MQZAO_ correspond to the keywords in the authorization list for the **GRTMQMAUT** and **RVKMQMAUT** commands for the particular entity. For example, MQZAO_BROWSE corresponds to the keyword `*BROWSE`; similarly, the keyword MQZAO_SET_ALL_CONTEXT corresponds to the keyword *SETALL and so on. These constants are defined in the header file cmqzc.h, which is supplied with the product.

MQI authorizations

An application is allowed to issue specific MQI calls and options only if the user identifier under which it is running (or whose authorizations it is able to assume) has been granted the relevant authorization.

Four MQI calls require authorization checks: MQCONN, MQOPEN, MQPUT1, and MQCLOSE.

For MQOPEN and MQPUT1, the authority check is made on the name of the object being opened, and not on the name, or names, resulting after a name has been resolved. For example, an application can be granted authority to open an alias queue without having authority to open the base queue to which the alias resolves. The rule is that the check is carried out on the first definition encountered during the process of name resolution that is not a queue-manager alias, unless the queue-manager alias definition is opened directly; that is, its name appears in the *ObjectName* field of the object descriptor. Authority is always needed for the particular object being opened; in some cases additional queue-independent authority, obtained through an authorization for the queue-manager object, is required.

Table 8, Table 9 on page 61, Table 10 on page 61, and Table 11 on page 62 summarize the authorizations needed for each call.

Note: You will find no mention of namelists, channels, client connection channels, listeners, services, or authentication information objects in these tables. This is because none of the authorizations apply to these objects, except for MQOO_INQUIRE, for which the same authorizations apply as for the other objects.

Table 8. Security authorization needed for MQCONN calls

Authorization required for:	Queue object (1 on page 62)	Process object	Queue manager object
MQCONN option	Not applicable	Not applicable	MQZAO_CONNECT

Table 9. Security authorization needed for MQOPEN calls

Authorization required for:	Queue object (1 on page 62)	Process object	Queue manager object
MQOO_INQUIRE	MQZAO_INQUIRE (2 on page 62)	MQZAO_INQUIRE (2 on page 62)	MQZAO_INQUIRE (2 on page 62)
MQOO_BROWSE	MQZAO_BROWSE	Not applicable	No check
MQOO_INPUT_*	MQZAO_INPUT	Not applicable	No check
MQOO_SAVE_ ALL_CONTEXT (3 on page 62)	MQZAO_INPUT	Not applicable	Not applicable
MQOO_OUTPUT (Normal queue) (4 on page 62)	MQZAO_OUTPUT	Not applicable	Not applicable
MQOO_PASS_ IDENTITY_CONTEXT (5 on page 62)	MQZAO_PASS_ IDENTITY_CONTEXT	Not applicable	No check
MQOO_PASS_ALL_ CONTEXT (5 on page 62, 6 on page 62)	MQZAO_PASS _ALL_CONTEXT	Not applicable	No check
MQOO_SET_ IDENTITY_CONTEXT (5 on page 62, 6 on page 62)	MQZAO_SET_ IDENTITY_CONTEXT	Not applicable	MQZAO_SET_ IDENTITY_CONTEXT (7 on page 62)
MQOO_SET_ ALL_CONTEXT (5 on page 62, 8 on page 62)	MQZAO_SET_ ALL_CONTEXT	Not applicable	MQZAO_SET_ ALL_CONTEXT (7 on page 62)
MQOO_OUTPUT (Transmission queue) (9 on page 62)	MQZAO_SET_ ALL_CONTEXT	Not applicable	MQZAO_SET_ ALL_CONTEXT (7 on page 62)
MQOO_SET	MQZAO_SET	Not applicable	No check
MQOO_ALTERNATE_ USER_AUTHORITY	(10 on page 62)	(10 on page 62)	MQZAO_ALTERNATE_ USER_AUTHORITY (10 on page 62, 11 on page 62)

Table 10. Security authorization needed for MQPUT1 calls

Authorization required for:	Queue object (1 on page 62)	Process object	Queue manager object
MQPMO_PASS_ IDENTITY_CONTEXT	MQZAO_PASS_ IDENTITY_CONTEXT (12 on page 62)	Not applicable	No check
MQPMO_PASS_ALL _CONTEXT	MQZAO_PASS_ ALL_CONTEXT (12 on page 62)	Not applicable	No check
MQPMO_SET_ IDENTITY_CONTEXT	MQZAO_SET_ IDENTITY_CONTEXT (12 on page 62)	Not applicable	MQZAO_SET_ IDENTITY_CONTEXT (7 on page 62)
MQPMO_SET_ ALL_CONTEXT	MQZAO_SET_ ALL_CONTEXT (12 on page 62)	Not applicable	MQZAO_SET_ ALL_CONTEXT (7 on page 62)
(Transmission queue) (9 on page 62)	MQZAO_SET_ ALL_CONTEXT	Not applicable	MQZAO_SET_ ALL_CONTEXT (7 on page 62)

Table 10. Security authorization needed for MQPUT1 calls (continued)

MQPMO_ALTERNATE_ USER_AUTHORITY	**(13)**	Not applicable	MQZAO_ALTERNATE_ USER_AUTHORITY **(11)**

Table 11. Security authorization needed for MQCLOSE calls

Authorization required for:	**Queue object (1)**	**Process object**	**Queue manager object**
MQCO_DELETE	MQZAO_DELETE **(14)**	Not applicable	Not applicable
MQCO_DELETE _PURGE	MQZAO_DELETE **(14)**	Not applicable	Not applicable

Notes for the tables:

1. If a model queue is being opened:
   - MQZAO_DISPLAY authority is needed for the model queue, in addition to the authority to open the model queue for the type of access for which you are opening.
   - MQZAO_CREATE authority is not needed to create the dynamic queue.
   - The user identifier used to open the model queue is automatically granted all the queue-specific authorities (equivalent to MQZAO_ALL) for the dynamic queue created.
2. Either the queue, process, namelist, or queue manager object is checked, depending on the type of object being opened.
3. MQOO_INPUT_* must also be specified. This is valid for a local, model, or alias queue.
4. This check is performed for all output cases, except the case specified in note 9.
5. MQOO_OUTPUT must also be specified.
6. MQOO_PASS_IDENTITY_CONTEXT is also implied by this option.
7. This authority is required for both the queue manager object and the particular queue.
8. MQOO_PASS_IDENTITY_CONTEXT, MQOO_PASS_ALL_CONTEXT, and MQOO_SET_IDENTITY_CONTEXT are also implied by this option.
9. This check is performed for a local or model queue that has a *Usage* queue attribute of MQUS_TRANSMISSION, and is being opened directly for output. It does not apply if a remote queue is being opened (either by specifying the names of the remote queue manager and remote queue, or by specifying the name of a local definition of the remote queue).
10. At least one of MQOO_INQUIRE (for any object type), or (for queues) MQOO_BROWSE, MQOO_INPUT_*, MQOO_OUTPUT, or MQOO_SET must also be specified. The check carried out is as for the other options specified, using the supplied alternate-user identifier for the specific-named object authority, and the current application authority for the MQZAO_ALTERNATE_USER_IDENTIFIER check.
11. This authorization allows any *AlternateUserId* to be specified.
12. An MQZAO_OUTPUT check is also carried out if the queue does not have a *Usage* queue attribute of MQUS_TRANSMISSION.
13. The check carried out is as for the other options specified, using the supplied alternate-user identifier for the named queue authority, and the current application authority for the MQZAO_ALTERNATE_USER_IDENTIFIER check.
14. The check is carried out only if both of the following are true:

- A permanent dynamic queue is being closed and deleted.
- The queue was not created by the MQOPEN that returned the object handle being used.

Otherwise, there is no check.

General notes:

1. The special authorization MQZAO_ALL_MQI includes all the following that are relevant to the object type:
   - MQZAO_CONNECT
   - MQZAO_INQUIRE
   - MQZAO_SET
   - MQZAO_BROWSE
   - MQZAO_INPUT
   - MQZAO_OUTPUT
   - MQZAO_PASS_IDENTITY_CONTEXT
   - MQZAO_PASS_ALL_CONTEXT
   - MQZAO_SET_IDENTITY_CONTEXT
   - MQZAO_SET_ALL_CONTEXT
   - MQZAO_ALTERNATE_USER_AUTHORITY
2. MQZAO_DELETE (see note 14 on page 62) and MQZAO_DISPLAY are classed as administration authorizations. They are not therefore included in MQZAO_ALL_MQI.
3. *No check* means that no authorization checking is carried out.
4. *Not applicable* means that authorization checking is not relevant to this operation. For example, you cannot issue an MQPUT call to a process object.

Administration authorizations

These authorizations allow a user to issue administration commands. This can be an MQSC command as an escape PCF message or as a PCF command itself. These methods allow a program to send an administration command as a message to a queue manager, for execution on behalf of that user.

Authorizations for MQSC commands in escape PCFs

This section summarizes the authorizations needed for each MQSC command contained in Escape PCF.

Not applicable means that authorization checking is not relevant to this operation.

The user ID under which the program that submits the command is running must also have the following authorities:

- MQZAO_CONNECT authority to the queue manager
- DISPLAY authority on the queue manager in order to perform PCF commands
- Authority to issue the MQSC commands within the text of the Escape PCF command

ALTER *object*

Object	Authorization required

Queue	MQZAO_CHANGE
Topic	MQZAO_CHANGE
Process	MQZAO_CHANGE
Queue manager	MQZAO_CHANGE
Namelist	MQZAO_CHANGE
Authentication information	MQZAO_CHANGE
Channel	MQZAO_CHANGE
Client connection channel	MQZAO_CHANGE
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

CLEAR *object*

Object	**Authorization required**
Queue	MQZAO_CLEAR
Topic	MQZAO_CLEAR
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	Not applicable
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

DEFINE *object* **NOREPLACE (1 on page 68)**

Object	**Authorization required**
Queue	MQZAO_CREATE **(2 on page 68)**
Topic	MQZAO_CREATE **(2 on page 68)**
Process	MQZAO_CREATE **(2 on page 68)**
Queue manager	Not applicable
Namelist	MQZAO_CREATE **(2 on page 68)**
Authentication information	MQZAO_CREATE **(2 on page 68)**
Channel	MQZAO_CREATE **(2 on page 68)**
Client connection channel	MQZAO_CREATE **(2 on page 68)**
Listener	MQZAO_CREATE **(2 on page 68)**
Service	MQZAO_CREATE **(2 on page 68)**

DEFINE *object* **REPLACE (1 on page 68, 3 on page 68)**

Object	**Authorization required**
Queue	MQZAO_CHANGE
Topic	MQZAO_CHANGE

Process	MQZAO_CHANGE
Queue manager	Not applicable
Namelist	MQZAO_CHANGE
Authentication information	MQZAO_CHANGE
Channel	MQZAO_CHANGE
Client connection channel	MQZAO_CHANGE
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

DELETE *object*

Object	**Authorization required**
Queue	MQZAO_DELETE
Topic	MQZAO_DELETE
Process	MQZAO_DELETE
Queue manager	Not applicable
Namelist	MQZAO_DELETE
Authentication information	MQZAO_DELETE
Channel	MQZAO_DELETE
Client connection channel	MQZAO_DELETE
Listener	MQZAO_DELETE
Service	MQZAO_DELETE

DISPLAY *object*

Object	**Authorization required**
Queue	MQZAO_DISPLAY
Topic	MQZAO_DISPLAY
Process	MQZAO_DISPLAY
Queue manager	MQZAO_DISPLAY
Namelist	MQZAO_DISPLAY
Authentication information	MQZAO_DISPLAY
Channel	MQZAO_DISPLAY
Client connection channel	MQZAO_DISPLAY
Listener	
Service	

PING CHANNEL

Object	**Authorization required**
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable

Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

RESET CHANNEL

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL_EXTENDED
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

RESOLVE CHANNEL

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL_EXTENDED
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

START CHANNEL

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable

Channel	MQZAO_CONTROL
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

START LISTENER

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	Not applicable
Client connection channel	Not applicable
Listener	MQZAO_CONTROL
Service	Not applicable

START SERVICE

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	Not applicable
Client connection channel	Not applicable
Listener	Not applicable
Service	MQZAO_CONTROL

STOP CHANNEL

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable

Listener	Not applicable
Service	Not applicable

STOP LISTENER

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	Not applicable
Client connection channel	Not applicable
Listener	MQZAO_CONTROL
Service	Not applicable

STOP SERVICE

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	Not applicable
Client connection channel	Not applicable
Listener	Not applicable
Service	MQZAO_CONTROL

Note:

1. For DEFINE commands, MQZAO_DISPLAY authority is also needed for the LIKE object if one is specified, or on the appropriate SYSTEM.DEFAULT.xxx object if LIKE is omitted.
2. The MQZAO_CREATE authority is not specific to a particular object or object type. Create authority is granted for all objects for a specified queue manager, by specifying an object type of QMGR on the **GRTMQAUT** command.
3. This applies if the object to be replaced already exists. If it does not, the check is as for DEFINE *object* NOREPLACE.

Authorizations for PCF commands

This section summarizes the authorizations needed for each PCF command.

No check means that no authorization checking is carried out; *Not applicable* means that authorization checking is not relevant to this operation.

The user ID under which the program that submits the command is running must also have the following authorities:

- MQZAO_CONNECT authority to the queue manager
- DISPLAY authority on the queue manager in order to perform PCF commands

The special authorization MQZAO_ALL_ADMIN includes the following authorizations:

- MQZAO_CHANGE
- MQZAO_CLEAR
- MQZAO_DELETE
- MQZAO_DISPLAY
- MQZAO_CONTROL
- MQZAO_CONTROL_EXTENDED

MQZAO_CREATE is not included as it is not specific to a particular object or object type

Change *object*

Object	**Authorization required**
Queue	MQZAO_CHANGE
Topic	MQZAO_CHANGE
Process	MQZAO_CHANGE
Queue manager	MQZAO_CHANGE
Namelist	MQZAO_CHANGE
Authentication information	MQZAO_CHANGE
Channel	MQZAO_CHANGE
Client connection channel	MQZAO_CHANGE
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

Clear *object*

Object	**Authorization required**
Queue	MQZAO_CLEAR
Topic	MQZAO_CLEAR
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	Not applicable
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

Copy *object* **(without replace) (1 on page 74)**

Object	**Authorization required**

Queue	MQZAO_CREATE **(2 on page 74)**
Topic	MQZAO_CREATE **(2 on page 74)**
Process	MQZAO_CREATE **(2 on page 74)**
Queue manager	Not applicable
NamelistMQZAO_CREATE	MQZAO_CREATE **(2 on page 74)**
Authentication information	MQZAO_CREATE **(2 on page 74)**
Channel	MQZAO_CREATE **(2 on page 74)**
Client connection channel	MQZAO_CREATE **(2 on page 74)**
Listener	MQZAO_CREATE **(2 on page 74)**
Service	MQZAO_CREATE **(2 on page 74)**

Copy *object* **(with replace) (1 on page 74, 4 on page 74)**

Object	**Authorization required**
Queue	MQZAO_CHANGE
Topic	MQZAO_CHANGE
Process	MQZAO_CHANGE
Queue manager	Not applicable
Namelist	MQZAO_CHANGE
Authentication information	MQZAO_CHANGE
Channel	MQZAO_CHANGE
Client connection channel	MQZAO_CHANGE
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

Create *object* **(without replace) (3 on page 74)**

Object	**Authorization required**
Queue	MQZAO_CREATE **(2 on page 74)**
Topic	MQZAO_CREATE **(2 on page 74)**
Process	MQZAO_CREATE **(2 on page 74)**
Queue manager	Not applicable
Namelist	MQZAO_CREATE **(2 on page 74)**
Authentication information	MQZAO_CREATE **(2 on page 74)**
Channel	MQZAO_CREATE **(2 on page 74)**
Client connection channel	MQZAO_CREATE **(2 on page 74)**
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

Create *object* **(with replace) (3 on page 74, 4 on page 74)**

Object	**Authorization required**
Queue	MQZAO_CHANGE
Topic	MQZAO_CHANGE

Process	MQZAO_CHANGE
Queue manager	Not applicable
Namelist	MQZAO_CHANGE
Authentication information	MQZAO_CHANGE
Channel	MQZAO_CHANGE
Client connection channel	MQZAO_CHANGE
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

Delete *object*

Object	**Authorization required**
Queue	MQZAO_DELETE
Topic	MQZAO_DELETE
Process	MQZAO_DELETE
Queue manager	MQZAO_DELETE
Namelist	MQZAO_DELETE
Authentication information	MQZAO_DELETE
Channel	MQZAO_DELETE
Client connection channel	MQZAO_DELETE
Listener	MQZAO_DELETE
Service	MQZAO_DELETE

Inquire *object*

Object	**Authorization required**
Queue	MQZAO_DISPLAY
Topic	MQZAO_DISPLAY
Process	MQZAO_DISPLAY
Queue manager	MQZAO_DISPLAY
Namelist	MQZAO_DISPLAY
Authentication information	MQZAO_DISPLAY
Channel	MQZAO_DISPLAY
Client connection channel	MQZAO_DISPLAY
Listener	MQZAO_DISPLAY
Service	MQZAO_DISPLAY

Inquire *object* **names**

Object	**Authorization required**
Queue	No check
Topic	No check
Process	No check
Queue manager	No check

Namelist	No check
Authentication information	No check
Channel	No check
Client connection channel	No check
Listener	No check
Service	No check

Ping Channel

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

Reset Channel

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL_EXTENDED
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

Reset Queue Statistics

Object	Authorization required
Queue	MQZAO_DISPLAY and MQZAO_CHANGE
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable

Channel	Not applicable
Client connection channel	Not applicable
Listener	
Service	

Resolve Channel

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL_EXTENDED
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

Start Channel

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

Stop Channel

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable

Listener	Not applicable
Service	Not applicable

Note:

1. For Copy commands, MQZAO_DISPLAY authority is also needed for the From object.
2. The MQZAO_CREATE authority is not specific to a particular object or object type. Create authority is granted for all objects for a specified queue manager, by specifying an object type of QMGR on the **GRTMQAUT** command.
3. For Create commands, MQZAO_DISPLAY authority is also needed for the appropriate SYSTEM.DEFAULT.* object.
4. This applies if the object to be replaced already exists. If it does not, the check is as for Copy or Create without replace.

Generic OAM profiles

OAM generic profiles enable you to set the authority a user has to many objects at once, rather than having to issue separate GRTMQMAUT commands against each individual object when it is created. Using generic profiles in the GRTMQMAUT command enables you to set a generic authority for all future objects created that fit that profile.

The rest of this section describes the use of generic profiles in more detail:
- "Using wildcard characters"
- "Profile priorities" on page 75

Using wildcard characters

What makes a profile generic is the use of special characters (wildcard characters) in the profile name. For example, the ? wildcard character matches any single character in a name. So, if you specify `ABC.?EF`, the authorization you give to that profile applies to any objects created with the names `ABC.DEF`, `ABC.CEF`, `ABC.BEF`, and so on.

The wildcard characters available are:

? Use the question mark (?) instead of any single character. For example, `AB.?D` would apply to the objects `AB.CD`, `AB.ED`, and `AB.FD`.

* Use the asterisk (*) as:

- A *qualifier* in a profile name to match any one qualifier in an object name. A qualifier is the part of an object name delimited by a period. For example, in `ABC.DEF.GHI`, the qualifiers are `ABC`, `DEF`, and `GHI`.

  For example, `ABC.*.JKL` would apply to the objects `ABC.DEF.JKL`, and `ABC.GHI.JKL`. (Note that it would **not** apply to `ABC.JKL`; * used in this context always indicates one qualifier.)
- A character within a qualifier in a profile name to match zero or more characters within the qualifier in an object name.

  For example, `ABC.DE*.JKL` would apply to the objects `ABC.DE.JKL`, `ABC.DEF.JKL`, and `ABC.DEGH.JKL`.

** Use the double asterisk (**) ***once*** in a profile name as:

- The entire profile name to match all object names. For example, if you use the keyword `OBJTYPE (*PRC)` to identify processes, then use ** as the profile name, you change the authorizations for all processes.
- As either the beginning, middle, or ending qualifier in a profile name to match zero or more qualifiers in an object name. For example, `**.ABC` identifies all objects with the final qualifier ABC.

Profile priorities

An important point to understand when using generic profiles is the priority that profiles are given when deciding what authorities to apply to an object being created. For example, suppose that you have issued the commands:

```
GRTMQMAUT OBJ(AB.*) OBJTYPE(*Q) USER(FRED) AUT(*PUT) MQMNAME(MYQMGR)
GRTMQMAUT OBJ(AB.C*) OBJTYPE(*Q) USER(FRED) AUT(*GET) MQMNAME(MYQMGR)
```

The first gives put authority to all queues for the principal `FRED` with names that match the profile AB.*; the second gives get authority to the same types of queue that match the profile AB.C*.

Suppose that you now create a queue called AB.CD. According to the rules for wildcard matching, either GRTMQMAUT could apply to that queue. So, does it have put or get authority?

To find the answer, you apply the rule that, whenever multiple profiles can apply to an object, **only the most specific applies**. The way that you apply this rule is by comparing the profile names from left to right. Wherever they differ, a non-generic character is more specific then a generic character. So, in the example above, the queue AB.CD has **get** authority (AB.C* is more specific than AB.*).

When you are comparing generic characters, the order of *specificity* is:
1. ?
2. *
3. **

Specifying the installed authorization service

The parameter `Service Component name` on GRTMQMAUT and RVKMQMAUT allows you to specify the name of the installed authorization service component.

Selecting F24 on the initial panel, followed by `F9=All parameters` on the next panel of either command, allows you to specify either the installed authorization component (*DFT) or the name of the required authorization service component specified in the Service stanza of the queue manager's qm.ini file.

DSPMQMAUT also has this extra parameter. This allows you to search all the installed authorization components (*DFT), or the specified authorization-service component name, for the specified object name, object type, and user

Working without authority profiles

You can work with authority profiles, as explained in "Working with authority profiles" on page 76, or without them, as explained here.

To use no authority profiles, use *NONE as an Authority parameter on GRTMQMAUT to create profiles without authority. This leaves any existing profiles unchanged.

On RVKMQMAUT, use *REMOVE as an Authority parameter to remove an existing authority profile.

Working with authority profiles

There are two commands associated with authority profiling:

- WRKMQMAUT
- WRKMQMAUTD

You can access these commands directly from the command line, or from the WRKMQM panel by:

1. Typing in the queue manager name and pressing the Enter key to access the WRKMQM results panel
2. Selecting F23=More options on this panel

Option 24 selects the results panel for the WRKMQMAUT command (see Figure 8 on page 77) and option 25 selects the WRKMQMAUTI command, which is used with the SSL bindings layer.

WRKMQMAUT

This command allows you to work with the authority data held in the authority queue. Figure 7 shows the input panel for this command.

Note: To run this command you must have *connect and *admdsp authority to the queue manager. However, to create or delete a profile, you need QMQMADM authority.

If you output the information to the screen, a list of authority profile names, together with their types, is displayed. If you print the output, you receive a detailed list of all the authority data, the registered users, and their authorities.

```
                    Work with MQ Authority (WRKMQMAUT)

Type choices, press Enter.

Object/Profile name  . . . . . .   *ALL

Object type  . . . . . . . . . .   *ALL          *Q, *PRC, *MQM, *NMLIST...
Output . . . . . . . . . . . . .   *             *, *PRINT
Message Queue Manager name . . .   *DFT



                                                                      Bottom

===> ____________________________________________________________________________
F3=Exit   F4=Prompt    F5=Refresh    F12=Cancel     F13= How to use this display
F24=More keys
```

Figure 7. Work with MQM Authority panel – input display

Entering an object or profile name on this panel, and pressing ENTER takes you to the results panel for WRKMQMAUT, shown in Figure 8.

```
                                Work with MQ Authority

Queue Manager Name . . :   *DFT

Type options, press Enter.
  4=Delete   12=Work with profile

Opt     Authority Profile Name                              Type
        SYSTEM.DEFAULT.NAMELIST                             *NMLIST
        SYSTEM.DEFAULT.PROCESS                              *PRC
        SYSTEM.DEFAULT.REMOTE.QUEUE                         *Q
        SYSTEM.MQSC.REPLY.QUEUE                             *Q
        SYSTEM.PENDING.DATA.QUEUE                           *Q









                                                                        Bottom
===>
F3=Exit   F4=Prompt      F5=Refresh   F6=Create   F9=Retrieve   F12=Cancel
F16=Repeat position to   F17=Position to          F21=Print
```

Figure 8. Work with MQM Authority panel – results display

If you select `4=Delete`, you go to a new panel from which you can confirm that you want to delete all the user names registered to the generic authority profile name you specify. This option runs RVKMQMAUT with the option *REMOVE for all the users, and applies **only** to generic profile names.

If you select `12=Work with profile` you go to the WRKMQMAUTD command results panel, as explained in "WRKMQMAUTD."

WRKMQMAUTD

This command allows you to display all the users registered with a particular authority profile name and object type. To run this command you must have `*connect` and `*admdsp` authority to the queue manager. However, to grant, run, create, or delete a profile you need QMQMADM authority.

Figure 9 on page 78 shows the input panel of the WRKMQMAUTD command.

```
                  Work with MQ Authority Data (WRKMQMAUTD)

Type choices, press Enter.

Object/Profile name  . . . . . .

Object type  . . . . . .                *Q, *PRC, *MQM, *NMLIST...
User name  . . . . . .                  Name, *PUBLIC, *ALL
Message Queue Manager name . . .



                                                                      Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 9. Work with MQM Authority Data input panel

Selecting F24=More keys from the initial input panel, followed by option F9=All Parameters displays the Service Component Name as for GRTMQMAUT and RVKMQMAUT.

Figure 10 shows the results panel of the WRKMQMAUTD command.

```
                        Work with MQ Authority Data

Queue Manager Name . . :   *DFT
Authority Profile Name :   SYSTEM.DEFAULT.PROCESS
Object Type  . . . . . :   *PRC

Type options, press Enter.
  2=Grant   3=Revoke   4=Delete   5=Display


Opt  UserName    GET  BROWSE  PUT  CONNECT  INQ  SET  ALTUSR
     MQUSER       X            X            X
     QMQMADM      X     X      X      X      X    X     X









                                                                      Bottom
===>
F3=Exit   F4=Prompt   F5=Refresh    F6=Create    F9=Retrieve
F11=Display Object Authorizations   F12=Cancel   F24=More keys
```

Figure 10. Work with MQM Authority Data output panel

Note: The F11=Display Object Authorizations key toggles between the following types of authorities:

- Object authorizations
- Context authorizations
- MQI authorizations

The options on the screen are:

2=Grant
Takes you to the GRTMQMAUT panel to add to the current authorities.

3=Revoke
Takes you to the RVKMQMAUT panel to remove some of the current definitions

4=Delete
Takes you to a panel that allows you to delete the authority data for specified users. This runs RVKMQMAUT with the option *REMOVE.

5=Display
Takes you to the existing DSPMQMAUT command

F6=Create
Takes you to the GRTMQMAUT panel that allows you to create a new profile authority record.

Object Authority Manager guidelines

Some operations are particularly sensitive; limit them to privileged users. For example,

- Accessing some special queues, such as transmission queues or the command queue SYSTEM.ADMIN.COMMAND.QUEUE
- Running programs that use full MQI context options
- Creating and copying application queues

Queue manager directories

The directories and libraries containing queues and other queue manager data are private to the product. Do not use standard operating system commands to grant or revoke authorizations to MQI resources.

Queues

The authority to a dynamic queue is based on, but is not necessarily the same as, that of the model queue from which it is derived.

For alias queues and remote queues, the authorization is that of the object itself, not the queue to which the alias or remote queue resolves. It is possible to authorize a user profile to access an alias queue that resolves to a local queue to which the user profile has no access permissions.

Limit the authority to create queues to privileged users. If you do not, users can bypass the normal access control simply by creating an alias.

Alternate-user authority

Alternate-user authority controls whether one user profile can use the authority of another user profile when accessing a WebSphere MQ object. This is essential where a server receives requests from a program and the server wants to ensure that the program has the required authority for the request. The server might have the required authority, but it needs to know whether the program has the authority for the actions it has requested.

For example:

- A server program running under user profile PAYSERV retrieves a request message from a queue that was put on the queue by user profile USER1.
- When the server program gets the request message, it processes the request and puts the reply back into the reply-to queue specified with the request message.
- Instead of using its own user profile (PAYSERV) to authorize opening the reply-to queue, the server can specify some other user profile, in this case, USER1. In this example, you can use alternate-user authority to control whether PAYSERV is allowed to specify USER1 as an alternate-user profile when it opens the reply-to queue.

The alternate-user profile is specified on the *AlternateUserId* field of the object descriptor.

Note: You can use alternate-user profiles on any WebSphere MQ object. Use of an alternate-user profile does not affect the user profile used by any other resource managers.

Context authority

Context is information that applies to a particular message and is contained in the message descriptor, MQMD, which is part of the message.

For descriptions of the message descriptor fields relating to context, see the WebSphere MQ Application Programming Reference.

For information about the context options, see the WebSphere MQ Application Programming Guide.

Remote security considerations

For remote security, consider:

Put authority
For security across queue managers you can specify the put authority that is used when a channel receives a message sent from another queue manager.

Specify the channel attribute PUTAUT as follows:

DEF Default user profile. This is the QMQM user profile under which the message channel agent is running.

CTX The user profile in the message context.

Transmission queues
Queue managers automatically put remote messages on a transmission queue; no special authority is required for this. However, putting a message directly on a transmission queue requires special authorization.

Channel exits
Channel exits can be used for added security.

For more information about remote security, see WebSphere MQ Intercommunication.

Channel command security

You can use SSL to protect channel commands.

Protecting channels with SSL

The Secure Sockets Layer (SSL) protocol provides out of the box channel security, with protection against eavesdropping, tampering, and impersonation. WebSphere MQ support for SSL enables you to specify, on the channel definition, that a particular channel uses SSL security. You can also specify details of the kind of security you want, such as the encryption algorithm you want to use.

SSL support in WebSphere MQ uses the queue manager *authentication information object* and various CL and MQSC commands and queue manager and channel parameters that define the SSL support required in detail.

The following CL commands support SSL:

WRKMQMAUTI
Work with the attributes of an authentication information object.

CHGMQMAUTI
Modify the attributes of an authentication information object.

CRTMQMAUTI
Create a new authentication information object.

CPYMQMAUTI
Create a new authentication information object by copying an existing one.

DLTMQMAUTI
Delete an authentication information object.

DSPMQMAUTI
Displays the attributes for a specific authentication information object.

For an overview of channel security using SSL, see WebSphere MQ Security.

For details of PCF commands associated with SSL, see WebSphere MQ Programmable Command Formats and Administration Interface.

Chapter 6. The WebSphere MQ dead-letter queue handler

A *dead-letter queue* (DLQ), sometimes referred to as an *undelivered-message queue*, is a holding queue for messages that cannot be delivered to their destination queues. Every queue manager in a network should have an associated DLQ.

Note: It is often preferable to avoid placing messages on a DLQ. For information about the use and avoidance of DLQs, see the WebSphere MQ Application Programming Guide.

Queue managers, message channel agents, and applications can put messages on the DLQ. All messages on the DLQ must be prefixed with a *dead-letter header* structure, MQDLH. Messages put on the DLQ by a queue manager or by a message channel agent always have an MQDLH. Always supply an MQDLH to applications putting messages on the DLQ. The *Reason* field of the MQDLH structure contains a reason code that identifies why the message is on the DLQ.

In all WebSphere MQ environments, there must be a routine that runs regularly to process messages on the DLQ. WebSphere MQ supplies a default routine, called the *dead-letter queue handler* (the DLQ handler), which you invoke using the STRMQMDLQ command. A user-written *rules table* supplies instructions to the DLQ handler, for processing messages on the DLQ. That is, the DLQ handler matches messages on the DLQ against entries in the rules table. When a DLQ message matches an entry in the rules table, the DLQ handler performs the action associated with that entry.

Invoking the DLQ handler

Use the STRMQMDLQ command to invoke the DLQ handler. You can name the DLQ you want to process and the queue manager you want to use in two ways:

- As parameters to STRMQMDLQ from the command prompt. For example:

```
STRMQMDLQ UDLMSGQ(ABC1.DEAD.LETTER.QUEUE) SRCMBR(QRULE) SRCFILE(library/QTXTSRC)
          MQMNAME(MY.QUEUE.MANAGER)
```

- In the rules table. For example:

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE)
```

Note: The rules table is a member within a source physical file that can take any name.

The above examples apply to the DLQ called ABC1.DEAD.LETTER.QUEUE, owned by the default queue manager.

If you do not specify the DLQ or the queue manager as shown above, the default queue manager for the installation is used along with the DLQ belonging to that queue manager.

The STRMQMDLQ command takes its input from the rules table.

You must be authorized to access both the DLQ itself, and any message queues to which messages on the DLQ are forwarded, in order to run the DLQ handler. You must also be authorized to assume the identity of other users, for the DLQ to put messages on queues with the authority of the user ID in the message context.

The DLQ handler rules table

The DLQ handler rules table defines how the DLQ handler is to process messages that arrive on the DLQ. There are two types of entry in a rules table:

- The first entry in the table, which is optional, contains *control data*.
- All other entries in the table are *rules* for the DLQ handler to follow. Each rule consists of a *pattern* (a set of message characteristics) that a message is matched against, and an *action* to be taken when a message on the DLQ matches the specified pattern. There must be at least one rule in a rules table.

Each entry in the rules table comprises one or more keywords.

Control data

This section describes the keywords that you can include in a control-data entry in a DLQ handler rules table. Note the following:

- The default value for a keyword, if any, is underlined.
- The vertical line (|) separates alternatives. You can specify only one of these.
- All keywords are optional.

INPUTQ (*QueueName***|'_')**
The name of the DLQ you want to process:

1. Any UDLMSGQ value (or *DFT) you specify as a parameter to the STRMQMDLQ command overrides any INPUTQ value in the rules table.
2. If you specify a blank UDLMSGQ value as a parameter to the STRMQMDLQ command, the INPUTQ value in the rules table is used.
3. If you specify a blank UDLMSGQ value as a parameter to the STRMQMDLQ command, and a blank INPUTQ value in the rules table, the system default dead-letter queue is used.

INPUTQM (*QueueManagerName***|'_')**
The name of the queue manager that owns the DLQ named on the INPUTQ keyword.

If you do not specify a queue manager, or you specify INPUTQM(' ') in the rules table, the system uses the default queue manager for the installation.

RETRYINT (*Interval***|60)**
The interval, in seconds, at which the DLQ handler should attempt to reprocess messages on the DLQ that could not be processed at the first attempt, and for which repeated attempts have been requested. By default, the retry interval is 60 seconds.

WAIT (YES|NO|*nnn***)**
Whether the DLQ handler should wait for further messages to arrive on the DLQ when it detects that there are no further messages that it can process.

YES Causes the DLQ handler to wait indefinitely.

NO Causes the DLQ handler to terminate when it detects that the DLQ is either empty or contains no messages that it can process.

nnn Causes the DLQ handler to wait for *nnn* seconds for new work to arrive before terminating, after it detects that the queue is either empty or contains no messages that it can process.

Specify WAIT (YES) for busy DLQs, and WAIT (NO) or WAIT (*nnn*) for DLQs that have a low level of activity. If the DLQ handler is allowed to terminate, re-invoke it using triggering.

You can supply the name of the DLQ as an input parameter to the STRMQMDLQ command, as an alternative to including control data in the rules table. If any value is specified both in the rules table and on input to the STRMQMDLQ command, the value specified on the STRMQMDLQ command takes precedence.

Note: If a control-data entry is included in the rules table, it ***must*** be the first entry in the table.

Rules (patterns and actions)

Here is an example rule from a DLQ handler rules table:

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
 ACTION (RETRY) RETRY (3)
```

This rule instructs the DLQ handler to make 3 attempts to deliver to its destination queue any persistent message that was put on the DLQ because MQPUT and MQPUT1 were inhibited.

This section describes the keywords that you can include in a rule. Note the following:

- The default value for a keyword, if any, is underlined. For most keywords, the default value is * (asterisk), which matches any value.
- The vertical line (|) separates alternatives. You can specify only one of these.
- All keywords except ACTION are optional.

This section begins with a description of the pattern-matching keywords (those against which messages on the DLQ are matched). It then describes the action keywords (those that determine how the DLQ handler is to process a matching message).

The pattern-matching keywords

The pattern-matching keywords are described below. Use them to specify values against which messages on the DLQ are matched. All pattern-matching keywords are optional.

APPLIDAT (*ApplIdentityData***|*)**
The *ApplIdentityData* value of the message on the DLQ, specified in the message descriptor, MQMD.

APPLNAME (*PutApplName***|*)**
The name of the application that issued the MQPUT or MQPUT1 call, as specified in the *PutApplName* field of the message descriptor, MQMD, of the message on the DLQ.

APPLTYPE (*PutApplType***|*)**
The *PutApplType* value specified in the message descriptor, MQMD, of the message on the DLQ.

DESTQ (*QueueName***|*)**
The name of the message queue for which the message is destined.

DESTQM (*QueueManagerName* **| *)**
The queue manager name for the message queue for which the message is destined.

FEEDBACK (*Feedback* **| *)**
When the *MsgType* value is MQMT_REPORT, *Feedback* describes the nature of the report.

You can use symbolic names. For example, you can use the symbolic name MQFB_COA to identify those messages on the DLQ that require confirmation of their arrival on their destination queues.

FORMAT (*Format* **| *)**
The name that the sender of the message uses to describe the format of the message data.

MSGTYPE (*MsgType* **| *)**
The message type of the message on the DLQ.

You can use symbolic names. For example, you can use the symbolic name MQMT_REQUEST to identify those messages on the DLQ that require replies.

PERSIST (*Persistence* **| *)**
The persistence value of the message. (The persistence of a message determines whether it survives restarts of the queue manager.)

You can use symbolic names. For example, you can use the symbolic name MQPER_PERSISTENT to identify those messages on the DLQ that are persistent.

REASON (*ReasonCode* **| *)**
The reason code that describes why the message was put to the DLQ.

You can use symbolic names. For example, you can use the symbolic name MQRC_Q_FULL to identify those messages placed on the DLQ because their destination queues were full.

REPLYQ (*QueueName* **| *)**
The reply-to queue name specified in the message descriptor, MQMD, of the message on the DLQ.

REPLYQM (*QueueManagerName* **| *)**
The queue manager name of the reply-to queue specified in the REPLYQ keyword.

USERID (*UserIdentifier* **| *)**
The user ID of the user who originated the message on the DLQ, as specified in the message descriptor, MQMD.

The action keywords

The action keywords are described below. Use them to describe how a matching message is processed.

ACTION (DISCARD | IGNORE | RETRY | FWD)
The action taken for any message on the DLQ that matches the pattern defined in this rule.

DISCARD
Causes the message to be deleted from the DLQ.

IGNORE
Causes the message to be left on the DLQ.

RETRY
Causes the DLQ handler to try again to put the message on its destination queue.

FWD Causes the DLQ handler to forward the message to the queue named on the FWDQ keyword.

You must specify the ACTION keyword. The number of attempts made to implement an action is governed by the RETRY keyword. The RETRYINT keyword of the control data controls the interval between attempts.

FWDQ (*QueueName***|&DESTQ|&REPLYQ)**
The name of the message queue to which the message is forwarded when you select the ACTION keyword.

QueueName
The name of a message queue. FWDQ(' ') is not valid.

&DESTQ
Take the queue name from the *DestQName* field in the MQDLH structure.

&REPLYQ
Take the queue name from the *ReplyToQ* field in the message descriptor, MQMD.

You can specify REPLYQ (?*) in the message pattern to avoid error messages, when a rule specifying FWDQ (&REPLYQ) matches a message with a blank *ReplyToQ* field.

FWDQM (*QueueManagerName***|&DESTQM|&REPLYQM|'_')**
The queue manager of the queue to which a message is forwarded.

QueueManagerName
The queue manager name for the queue to which the message is forwarded when you select the ACTION (FWD) keyword.

&DESTQM
Take the queue manager name from the *DestQMgrName* field in the MQDLH structure.

&REPLYQM
Take the queue manager name from the *ReplyToQMgr* field in the message descriptor, MQMD.

' ' FWDQM(' '), which is the default value, identifies the local queue manager.

HEADER (YES|NO)
Whether the MQDLH should remain on a message for which ACTION (FWD) is requested. By default, the MQDLH remains on the message. The HEADER keyword is not valid for actions other than FWD.

PUTAUT (DEF|CTX)
The authority with which messages should be put by the DLQ handler:

DEF Puts messages with the authority of the DLQ handler itself.

CTX Causes the messages to be put with the authority of the user ID in the message context. You must be authorized to assume the identity of other users, if you specify PUTAUT (CTX).

RETRY (*RetryCount* **| 1)**
The number of times, in the range 1–999 999 999, to attempt an action (at the interval specified on the RETRYINT keyword of the control data).

Note: The count of attempts made by the DLQ handler to implement any particular rule is specific to the current instance of the DLQ handler; the count does not persist across restarts. If you restart the DLQ handler, the count of attempts made to apply a rule is reset to zero.

Rules table conventions

The rules table must adhere to the following conventions regarding its syntax, structure, and contents:

- A rules table must contain at least one rule.
- Keywords can occur in any order.
- A keyword can be included once only in any rule.
- Keywords are not case sensitive.
- A keyword and its parameter value must be separated from other keywords by at least one blank or comma.
- Any number of blanks can occur at the beginning or end of a rule, and between keywords, punctuation, and values.
- Each rule must begin on a new line.
- For portability, the significant length of a line must not be greater than 72 characters.
- Use the plus sign (+) as the last non-blank character on a line to indicate that the rule continues from the first non-blank character in the next line. Use the minus sign (-) as the last non-blank character on a line to indicate that the rule continues from the start of the next line. Continuation characters can occur within keywords and parameters.

  For example:

  ```
  APPLNAME('ABC+
    D')
  ```

  results in 'ABCD'.

  ```
  APPLNAME('ABC-
    D')
  ```

  results in 'ABC D'.
- Comment lines, which begin with an asterisk (*), can occur anywhere in the rules table.
- Blank lines are ignored.
- Each entry in the DLQ handler rules table comprises one or more keywords and their associated parameters. The parameters must follow these syntax rules:
  - Each parameter value must include at least one significant character. The delimiting quotation marks in quoted values are not considered significant. For example, these parameters are valid:

    ```
    FORMAT('ABC')       3 significant characters
    FORMAT(ABC)         3 significant characters
    FORMAT('A')         1 significant character
    FORMAT(A)           1 significant character
    FORMAT(' ')         1 significant character
    ```

These parameters are invalid because they contain no significant characters:

```
FORMAT('')
FORMAT( )
FORMAT()
FORMAT
```

- Wildcard characters are supported. You can use the question mark (?) in place of any single character, except a trailing blank. You can use the asterisk (*) in place of zero or more adjacent characters. The asterisk (*) and the question mark (?) are ***always*** interpreted as wildcard characters in parameter values.
- You cannot include wildcard characters in the parameters of these keywords: ACTION, HEADER, RETRY, FWDQ, FWDQM, and PUTAUT.
- Trailing blanks in parameter values, and in the corresponding fields in the message on the DLQ, are not significant when performing wildcard matches. However, leading and embedded blanks within strings in quotation marks are significant to wildcard matches.
- Numeric parameters cannot include the question mark (?) wildcard character. You can include the asterisk (*) in place of an entire numeric parameter, but the asterisk cannot be included as part of a numeric parameter. For example, these are valid numeric parameters:

`MSGTYPE(2)`	Only reply messages are eligible
`MSGTYPE(*)`	Any message type is eligible
`MSGTYPE('*')`	Any message type is eligible

  However, `MSGTYPE('2*')` is not valid, because it includes an asterisk (*) as part of a numeric parameter.
- Numeric parameters must be in the range 0–999 999 999. If the parameter value is in this range, it is accepted, even if it is not currently valid in the field to which the keyword relates. You can use symbolic names for numeric parameters.
- If a string value is shorter than the field in the MQDLH or MQMD to which the keyword relates, the value is padded with blanks to the length of the field. If the value, excluding asterisks, is longer than the field, an error is diagnosed. For example, these are all valid string values for an 8-character field:

`'ABCDEFGH'`	8 characters
`'A*C*E*G*I'`	5 characters excluding asterisks
`'*A*C*E*G*I*K*M*O*'`	8 characters excluding asterisks

- Strings that contain blanks, lowercase characters, or special characters other than period (.), forward slash (/), underscore (_), and percent sign (%) must be enclosed in single quotation marks. Lowercase characters not enclosed in quotation marks are folded to uppercase. If the string includes a quotation, two single quotation marks must be used to denote both the beginning and the end of the quotation. When the length of the string is calculated, each occurrence of double quotation marks is counted as a single character.

Processing the rules table

The DLQ handler searches the rules table for a rule whose pattern matches a message on the DLQ. The search begins with the first rule in the table, and continues sequentially through the table. When a rule with a matching pattern is found, the rules table attempts the action from that rule. The DLQ handler increments the retry count for a rule by 1 whenever it attempts to apply that rule. If the first attempt fails, the attempt is repeated until the count of attempts made matches the number specified on the RETRY keyword. If all attempts fail, the DLQ handler searches for the next matching rule in the table.

This process is repeated for subsequent matching rules until an action is successful. When each matching rule has been attempted the number of times specified on its RETRY keyword, and all attempts have failed, ACTION (IGNORE) is assumed. ACTION (IGNORE) is also assumed if no matching rule is found.

Note:

1. Matching rule patterns are sought only for messages on the DLQ that begin with an MQDLH. Messages that do not begin with an MQDLH are reported periodically as being in error, and remain on the DLQ indefinitely.
2. All pattern keywords can default, so that a rule can consist of an action only. Note, however, that action-only rules are applied to all messages on the queue that have MQDLHs and that have not already been processed in accordance with other rules in the table.
3. The rules table is validated when the DLQ handler starts, and errors flagged at that time. (Error messages issued by the DLQ handler are described in WebSphere MQ Messages.) You can make changes to the rules table at any time, but those changes do not come into effect until the DLQ handler is restarted.
4. The DLQ handler does not alter the content of messages, of the MQDLH, or of the message descriptor. The DLQ handler always puts messages to other queues with the message option MQPMO_PASS_ALL_CONTEXT.
5. Consecutive syntax errors in the rules table might not be recognized, because the validation of the rules table is designed to eliminate the generation of repetitive errors.
6. The DLQ handler opens the DLQ with the MQOO_INPUT_AS_Q_DEF option.
7. Multiple instances of the DLQ handler can run concurrently against the same queue, using the same rules table. However, it is more usual for there to be a one-to-one relationship between a DLQ and a DLQ handler.

Ensuring that all DLQ messages are processed

The DLQ handler keeps a record of all messages on the DLQ that have been seen but not removed. If you use the DLQ handler as a filter to extract a small subset of the messages from the DLQ, the DLQ handler still keeps a record of those messages on the DLQ that it did not process. Also, the DLQ handler cannot guarantee that new messages arriving on the DLQ will be seen, even if the DLQ is defined as first-in first-out (FIFO). If the queue is not empty, the DLQ is periodically re-scanned to check all messages.

For these reasons, try to ensure that the DLQ contains as few messages as possible. If messages that cannot be discarded or forwarded to other queues (for whatever

reason) are allowed to accumulate on the queue, the workload of the DLQ handler increases and the DLQ itself is in danger of filling up.

You can take specific measures to enable the DLQ handler to empty the DLQ. For example, try not to use ACTION (IGNORE), which simply leaves messages on the DLQ. (Remember that ACTION (IGNORE) is assumed for messages that are not explicitly addressed by other rules in the table.) Instead, for those messages that you would otherwise ignore, use an action that moves the messages to another queue. For example:

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

Similarly, make the final rule in the table a catchall to process messages that have not been addressed by earlier rules in the table. For example, the final rule in the table could be something like this:

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

This causes messages that fall through to the final rule in the table to be forwarded to the queue REALLY.DEAD.QUEUE, where they can be processed manually. If you do not have such a rule, messages are likely to remain on the DLQ indefinitely.

An example DLQ handler rules table

Here is an example rules table that contains a single control-data entry and several rules:

```
*****************************************************************************
*        An example rules table for the STRMQMDLQ command                   *
*****************************************************************************
* Control data entry
* ------------------
* If no queue manager name is supplied as an explicit parameter to
* STRMQMDLQ, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to STRMQMDLQ,
* use the DLQ defined for the local queue manager.
*
inputqm(' ')  inputq(' ')

* Rules
* -----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.

* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation is always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
```

```
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
  ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never does things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
  action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
* where the problem is investigated.

REPLYQM(CCCC.*) +
  ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it must be able
* to cope with the message being lost, so we can afford to
* discard the message.

PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)

* For performance and efficiency reasons, we like to keep
* the number of messages on the DLQ small.
* If we receive a message that has not been processed by
* an earlier rule in the table, we assume that it
* requires manual intervention to resolve the problem.
* Some problems are best solved at the node where the
* problem was detected, and others are best solved where
* the message originated. We do not have the message origin,
* but we can use the REPLYQM to identify a node that has
* some interest in this message.
* Attempt to put the message onto a manual intervention
* queue at the appropriate node. If this fails,
* put the message on the manual intervention queue at
* this node.

REPLYQM('?*') +
  ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

Chapter 7. Backup, recovery, and restart

WebSphere MQ for i5/OS uses the i5/OS journaling support to help its backup and restore strategy. You must be familiar with standard i5/OS backup and recovery methods, and with the use of journals and their associated journal receivers on i5/OS, before reading this section. For information on these topics, see *i5/OS Backup and Recovery*.

To understand the backup and recovery strategy, you first need to understand how WebSphere MQ for i5/OS organizes its data in the i5/OS file system and the integrated file system (IFS)

WebSphere MQ for i5/OS holds its data in an individual library for each queue manager, and in stream files in the IFS file system.

The queue manager specific libraries contain journals, journal receivers, and objects required to control the work management of the queue manager. The IFS directories and files contain WebSphere MQ configuration files, the descriptions of WebSphere MQ objects, and the data they contain.

Every change to these objects, that is recoverable across a system failure, is recorded in a journal *before* it is applied to the appropriate object. This has the effect that such changes can be recovered by replaying the information recorded in the journal.

WebSphere MQ for i5/OS journals

WebSphere MQ for i5/OS uses journals in its operation to control updates to local objects. Each queue manager library contains a journal for that queue manager, which has the name `QMGRLIB/AMQAJRN`, where `QMGRLIB` is the name of the queue manager library.

`QMGRLIB` takes the name `QM`, followed by the name of the queue manager in a unique form. For example, a queue manager named `TEST` has a journal receiver library named `QMTEST`.

These journals have associated journal receivers that contain the information being journaled. These receivers are objects to which information can only be appended and will fill up eventually.

Journal receivers also use up valuable disk space with out-of-date information. However, you can place the information in permanent storage to minimize this problem. One journal receiver is attached to the journal at any particular time. If the journal receiver reaches its predetermined threshold size, it is detached and replaced by a new journal receiver. You can specify the threshold of journal receivers when you create a queue manager using **CRTMQM** and the THRESHOLD parameter.

The journal receivers associated with the local WebSphere MQ for i5/OS journal exist in each queue manager library, and adopt a naming convention as follows:

```
AMQArnnnnn
```

where

nnnnn
is decimal 00000 to 99999

r is decimal 0 to 9

The sequence of the journals is based on date. However, the naming of the next journal is based on the following rules:

1. AMQArnnnnn goes to AMQAr(nnnnn+1), and nnnnn wraps when it reaches 99999. For example, AMQA000000 goes to AMQA000001, and AMQA999999 goes to AMQA000000.
2. If a journal with a name generated by rule 1 already exists, the message CPI70E3 is sent to the QSYSOPR message queue and automatic receiver switching stops.

   The currently-attached receiver continues to be used until you investigate the problem and manually attach a new receiver.
3. If no new name is available in the sequence (that is, all possible journal names are on the system) you need to do both of the following:
   a. Delete journals no longer needed (see "Journal management" on page 99).
   b. Record the journal changes into the latest journal receiver using (RCDMQMIMG) and then repeat the previous step. This allows the old journal receiver names to be reused.

The AMQAJRN journal uses the MNGRCV(*SYSTEM) option to enable the operating system to automatically change journal receivers when the threshold is reached. For more information on how the system manages receivers, see *i5/OS Backup and Recovery*.

The journal receiver's default threshold value is 100 000 KB. You can set this to a larger value when you create the queue manager. The initial value of the LogReceiverSize attribute is written to the LogDefaults stanza of the mqs.ini file.

When a journal receiver extends beyond its specified threshold, the receiver is detached and a new journal receiver is created, inheriting attributes from the previous receiver. Changes to the LogReceiverSize or LogASP attributes after a queue manager has been created are ignored when the system automatically attaches a new journal receiver

See Chapter 9, "Configuring WebSphere MQ," on page 125 for further details on configuring the system.

If you need to change the size of journal receivers after the queue manager has been created, create a new journal receiver and set its owner to QMQM using the following commands:

```
CRTJRNRCV JRNRCV(QMGRLIB/AMQAnnnnnn) THRESHOLD(xxxxxx) +
   TEXT('MQM LOCAL JOURNAL RECEIVER')
CHGOBJOWN OBJ(QMGRLIB/AMQAnnnnnn) OBJTYPE(*JRNRCV) NEWOWN(QMQM)
```

where

QmgrLib
Is the name of your queue manager library

nnnnnnn
Is the next journal receiver in the naming sequence described

xxxxxx Is the new receiver threshold (in KB)

> **Note:** The maximum size of the receiver is governed by the operating system. To check this value look at the THRESHOLD keyword on the CRTJRNRCV command.

Now attach the new receiver to the AMQAJRN journal with the command:

```
CHGJRN JRN(QMGRLIB/AMQAJRN) JRNRCV(QMGRLIB/AMQAnnnnnn)
```

See "Journal management" on page 99 for details on how to manage these journal receivers.

WebSphere MQ for i5/OS journal usage

Persistent updates to message queues happen in two stages. The records representing the update are first written to the journal, then the queue file is updated.

The journal receivers can therefore become more up-to-date than the queue files. To ensure that restart processing begins from a consistent point, WebSphere MQ uses checkpoints.

A checkpoint is a point in time when the record described in the journal is the same as the record in the queue. The checkpoint itself consists of the series of journal records needed to restart the queue manager. For example, the state of all transactions (that is, units of work) active at the time of the checkpoint.

Checkpoints are generated automatically by WebSphere MQ. They are taken when the queue manager starts and shuts down, and after a certain number of operations are logged.

You can force a queue manager to take a checkpoint by issuing the RCDMQMIMG command against all objects on a queue manager and displaying the results, as follows:

```
RCDMQMIMG  OBJ(*ALL)  OBJTYPE(*ALL)  MQMNAME(<Q_MGR_NAME>)  DSPJRNDTA(*YES)
```

As the queues handle further messages, the checkpoint record becomes inconsistent with the current state of the queues.

When WebSphere MQ is restarted, it locates the latest checkpoint record in the log. This information is held in the checkpoint file that is updated at the end of every checkpoint. The checkpoint record represents the most recent point of consistency between the log and the data. The data from this checkpoint is used to rebuild the queues as they existed at the checkpoint time. When the queues are recreated, the log is then played forward to bring the queues back to the state they were in before system failure or close down.

To understand how WebSphere MQ uses the journal, consider the case of a local queue called `TESTQ` in the queue manager `TEST`. This is represented by the IFS file:

```
/QIBM/UserData/mqm/qmgrs/TEST/queues
```

If a specified message is put on this queue, and then retrieved from the queue, the actions that take place are shown in Figure Figure 11 on page 96.

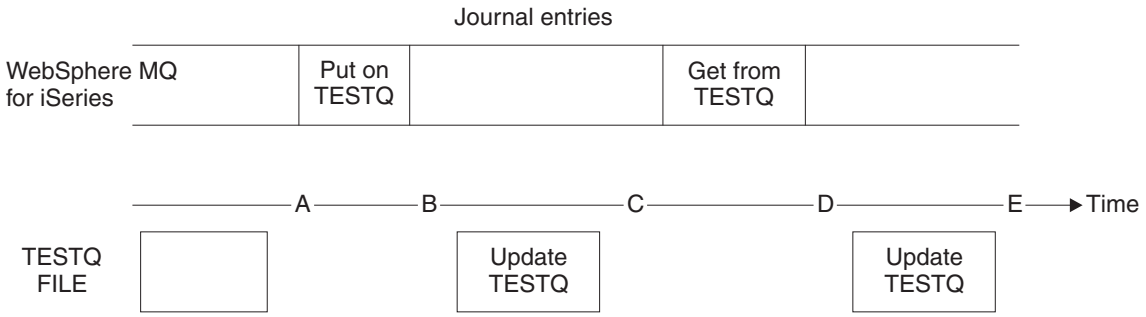

Figure 11. Sequence of events when updating MQM objects

The five points, A through E, shown in the diagram represent points in time that define the following states:

A The IFS file representation of the queue is consistent with the information contained in the journal.

B A journal entry is written to the journal defining a `Put` operation on the queue.

C The appropriate update is made to the queue.

D A journal entry is written to the journal defining a `Get` operation from the queue.

E The appropriate update is made to the queue.

The key to the recovery capabilities of WebSphere MQ for i5/OS is that the user can save the IFS file representation of TESTQ as at time `A`, and subsequently recover the IFS file representation of TESTQ as at time `E`, simply by restoring the saved object and replaying the entries in the journal from time `A` onwards.

This strategy is used by WebSphere MQ for i5/OS to recover persistent messages after system failure. WebSphere MQ remembers a particular entry in the journal receivers, and ensures that on startup it replays the entries in the journals from this point onwards. This startup entry is periodically recalculated so that WebSphere MQ only has to perform the minimum necessary replay on the next startup.

WebSphere MQ provides individual recovery of objects. All persistent information relating to an object is recorded in the local WebSphere MQ for i5/OS journals. Any WebSphere MQ object that becomes damaged or corrupt can be completely rebuilt from the information held in the journal.

For more information on how the system manages receivers, see *i5/OS Backup and Recovery*.

Media images

A WebSphere MQ object of long duration can represent a large number of journal entries, going back to the point at which it was created. To avoid this overhead, WebSphere MQ for i5/OS has the concept of a *media image* of an object.

This media image is a complete copy of the WebSphere MQ object recorded in the journal. If an image of an object is taken, the object can be rebuilt by replaying

journal entries from this image onwards. The entry in the journal that represents the replay point for each WebSphere MQ object is referred to as its *media recovery entry*. WebSphere MQ keeps track of the:

- Media recovery entry for each queue manager object.
- Oldest entry from within this set (see error message AMQ7462 in WebSphere MQ Messages for details.

Images of the *CTLG object and the *MQM object are taken regularly because these objects are crucial to queue manager restart.

Images of other objects are taken when convenient. By default, images of **all** objects are taken when a queue manager is shut down using **ENDMQM** with parameter ENDCCTJOB(*YES). This operation can take a considerable amount of time for very large queue managers. If you need to shut down quickly, specify parameter RCDMQMIMG(*NO) with ENDCCTJOB(*YES). In such cases, you are recommended to record a complete media image in the journals after the queue manager has been restarted, using the following command:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(<Q_MGR_NAME>)
```

WebSphere MQ automatically records an image of an object, if it finds a convenient point at which an object can be compactly described by a small entry in the journal. However, this might never happen for some objects, for example, queues that consistently contain large numbers of messages.

Rather than allow the date of the oldest media recovery entry to continue for an unnecessarily long period, use the WebSphere MQ command `RCDMQMIMG`, which enables you to take an image of selected objects manually.

Recovery from media images

WebSphere MQ automatically recovers some objects from their media image if it is found that they are corrupt or damaged. In particular, this applies to the special *MQM and *CTLG objects as part of the normal queue manager startup. If any syncpoint transaction was incomplete at the time of the last shutdown of the queue manager, any queue affected is also recovered automatically, in order to complete the startup operation.

You must recover other objects manually, using the WebSphere MQ command `RCRMQMOBJ`. This command replays the entries in the journal to recreate the WebSphere MQ object. Should a WebSphere MQ object become damaged, the only valid actions are to delete it or recreate it by this method. Note, however, that nonpersistent messages cannot be recovered in this fashion.

Checkpoints

As described above, checkpoints are taken at various times to provide a known consistent start point for recovery. The checkpoint process AMQALMPX is responsible for taking the checkpoint at the following points:

- Queue manager startup (STRMQM).
- Queue manager shutdown (ENDMQM).
- After a period of time has elapsed since the last checkpoint (the default period is 30 minutes) and a minimum number of log records have been written since the previous checkpoint (the default value is 100).
- After a number of log records have been written. The default value is 10 000.

- After the journal threshold size has been exceeded and a new journal receiver has been automatically created.
- When a full media image is taken with:

  ```
  RCDMQMIMG  OBJ(*ALL)  OBJTYPE(*ALL)  MQMNAME(<Q_MGR_NAME>)  DSPJRNDTA(*YES)
  ```

Backups of WebSphere MQ for i5/OS data

For each queue manager, there are two types of WebSphere MQ backup to consider:

- Data and journal backup.

  To ensure that both sets of data are consistent, do this only after shutting down the queue manager.
- Journal backup.

  You can do this while the queue manager is active.

For both methods, you need to find the names of the queue manager IFS directory and the queue manager library. You can find these in the WebSphere MQ configuration file (mqs.ini). For more information, see "The QueueManager stanza" on page 129.

Use the procedures below to do both types of backup:

Data and journal backup of a particular queue manager

Note: Do not use a save-while-active request when the queue manager is running. Such a request cannot complete unless all commitment definitions with pending changes are committed or rolled back. If this command is used when the queue manager is active, the channel connections might not end normally. Always use the procedure described below.

1. Create an empty journal receiver, using the command:

   ```
   CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
   ```
2. Use the RCDMQMIMG command to record an MQM image for all WebSphere MQ objects, and then force a checkpoint using the command:

   ```
   RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
   ```
3. End channels and ensure that the queue manager is not running. If your queue manager is running, stop it with the ENDMQM command.
4. Backup the queue manager library by issuing the following command:

   ```
   SAVLIB LIB(QMTEST)
   ```
5. Back up the queue manager IFS directories by issuing the following command:

   ```
   SAV DEV(...)  OBJ(('/QIBM/UserData/mqm/qmgrs/test'))
   ```

Journal backup of a particular queue manager

Because all relevant information is held in the journals, as long as you perform a full save at some time, partial backups can be performed by saving the journal receivers. These record all changes since the time of the full backup and are performed by issuing the following commands:

1. Create an empty journal receiver, using the command:

   ```
   CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
   ```

2. Use the RCDMQMIMG command to record an MQM image for all WebSphere MQ objects, and then force a checkpoint using the command:

   ```
   RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
   ```

3. Save the journal receivers using the command:

   ```
   SAVOBJ OBJ(AMQ*) LIB(QMTEST) OBJTYPE(*JRNRCV) .........
   ```

A simple backup strategy is to perform a full backup of the WebSphere MQ libraries every week, and perform a daily journal backup. This, of course, depends on how you have set up your backup strategy for your enterprise.

Journal management

As part of your backup strategy, take care of your journal receivers. It is useful to remove journal receivers from the WebSphere MQ libraries, in order to:

1. Release space; this applies to all journal receivers
2. Improve the performance when starting (STRMQM)
3. Improve the performance of recreating objects (RCRMQMOBJ)

Before deleting a journal receiver, be sure that:

1. You have a backup copy.
2. You no longer need the journal receiver.

Journal receivers can be removed from the queue manager library *after* they have been detached from the journals and saved, provided that they are available for restoration if needed for a recovery operation.

The concept of journal management is shown in Figure 12 on page 100.

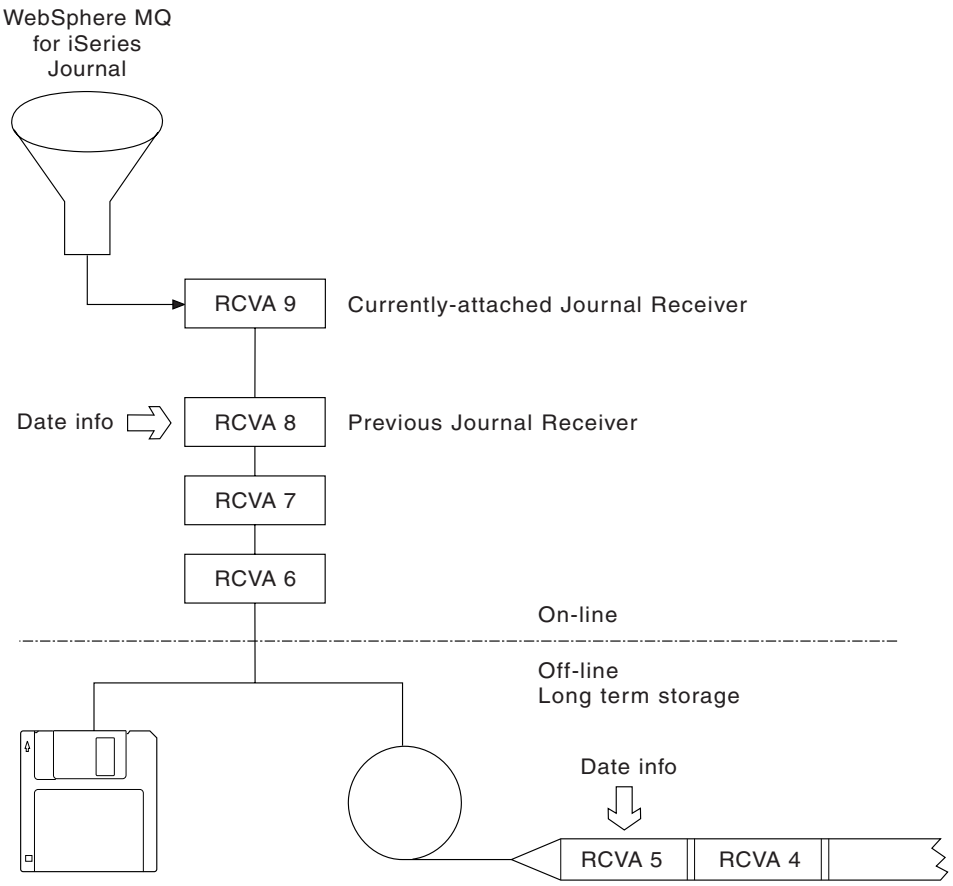

Figure 12. WebSphere MQ for i5/OS journaling

It is important to know how far back in the journals WebSphere MQ is likely to need to go, in order to determine when a journal receiver that has been backed up can be removed from the queue manager library, and when the backup itself can be discarded.

To help determine this time, WebSphere MQ issues two messages to the queue manager message queue (QMQMMSG in the queue manager library) when:

- It starts up
- It changes a local journal receiver
- You use RCDMQIMG to force a checkpoint

These messages are:

AMQ7460
Startup recovery point. This message defines the date and time of the startup entry from which WebSphere MQ replays the journal in the event of a startup recovery pass. If the journal receiver that contains this record is available in the WebSphere MQ libraries, this message also contains the name of the journal receiver containing the record.

AMQ7462
Oldest media recovery entry. This message defines the date and time of the oldest entry to use to recreate an object from its media image.

The journal receiver identified is the oldest one required. Any other WebSphere MQ journal receivers with older creation dates are no longer

needed. If only stars are displayed, you need to restore backups from the date indicated to determine which is the oldest journal receiver.

When these messages are logged, WebSphere MQ also writes a user space object to the queue manager library that contains only one entry: the name of the oldest journal receiver that needs to be kept on the system. This user space is called AMQJRNINF, and the data is written in the format:

```
JJJJJJJJJJLLLLLLLLLLYYYYMMDDHHMMSSmmm
```

where:

JJJJJJJJJJ
Is the oldest receiver name that WebSphere MQ still needs.

LLLLLLLLLL
Is the journal receiver library name.

YYYY Is the year of the oldest journal entry that WebSphere MQ needs.

MM Is the month of the oldest journal entry that WebSphere MQ needs.

DD Is the day of the oldest journal entry that WebSphere MQ needs.

HH Is the hour of the oldest journal entry that WebSphere MQ needs.

SS Is the seconds of the oldest journal entry that WebSphere MQ needs.

mmm Is the milliseconds of the oldest journal entry that WebSphere MQ needs.

When the oldest journal receiver has been deleted from the system, this user space contains asterisks (*) for the journal receiver name.

Note: Periodically performing `RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES)` can save startup time for WebSphere MQ and reduce the number of local journal receivers you need to save and restore for recovery.

WebSphere MQ for i5/OS does not refer to the journal receivers unless it is performing a recovery pass either for startup, or for recreating an object. If it finds that a journal it requires is not present, it issues message `AMQ7432` to the queue manager message queue (QMQMMSG), reporting the time and date of the journal entry it requires to complete the recovery pass.

If this happens, restore all journal receivers that were detached after this date from the backup, in order to allow the recovery pass to succeed.

Keep the journal receiver that contains the startup entry, and any subsequent journal receivers, available in the queue manager library.

Keep the journal receiver containing the oldest `Media Recovery Entry`, and any subsequent journal receivers, available at all times, and either present in the queue manager library or backed-up.

When you force a checkpoint:

- If the journal receiver named in AMQ7460 is not advanced, this indicates that there is an incomplete unit of work that needs to be committed or rolled back.
- If the journal receiver named in AMQ7462 is not advanced, this indicates that there are one or more damaged objects.

Restoring a complete queue manager (data and journals)

If you need to recover one or more WebSphere MQ queue managers from a backup, perform the following steps.

1. Quiesce the WebSphere MQ queue managers.
2. Locate your latest backup set, consisting of your most recent full backup and subsequently backed up journal receivers.
3. Perform a RSTLIB operation, from the full backup, to restore the WebSphere MQ data libraries to their state at the time of the full backup, by issuing the following commands:

   ```
   RSTLIB LIB(QMQRLIB1) .........
   RSTLIB LIB(QMQRLIB2) .........
   ```

   If a journal receiver was partially saved in one journal backup, and fully saved in a subsequent backup, restore only the fully saved one. Restore journals individually, in chronological order.
4. Perform an RST operation to restore the WebSphere MQ IFS directories to the IFS file system, using the following command:

   ```
   RST DEV(...) OBJ(('/QIBM/UserData/mqm/qmgrs/testqm'))  ...
   ```
5. Start the message queue manager. This replays all journal records written since the full backup and restores all the WebSphere MQ objects to the consistent state at the time of the journal backup.

If you want to restore a complete queue manager on a different machine, use the procedure below to restore everything from the queue manager library. (We use TEST as the sample queue manager name.)

1. `CRTMQM TEST`
2. `DLTLIB LIB(QMTEST)`
3. `RSTLIB SAVLIB(QMTEST) DEV(*SAVF) SAVF(QMGRLIBSAV)`
4. Delete the following IFS files:

   ```
   /QIBM/UserData/mqm/qmgrs/TEST/QMQMCHKPT
   /QIBM/UserData/mqm/qmgrs/TEST/qmanager/QMQMOBJCAT
   /QIBM/UserData/mqm/qmgrs/TEST/qmanager/QMANAGER
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.AUTH.DATA.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CHANNEL.INITQ/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.COMMAND.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/
   SYSTEM.CLUSTER.REPOSITORY.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.TRANSMIT.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.PENDING.DATA.QUEUE/q
   /QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.ADMIN.COMMAND.QUEUE/q
   ```
5. `STRMQM TEST`
6. `RCRMQMOBJ OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(TEST)`

Restoring journal receivers for a particular queue manager

The most common action is to restore a backed-up journal receiver to a queue manager library, if a receiver that has been removed is needed again for a subsequent recovery function.

This is a simple task, and requires the journal receivers to be restored using the standard i5/OS RSTOBJ command:

```
RSTOBJ OBJ(QMQMDATA/AMQA000005) OBJTYPE(*JRNRCV) .........
```

A series of journal receivers might need to be restored, rather than a single receiver. For example, AMQA000007 is the oldest receiver in the WebSphere MQ libraries, and both AMQA000005 and AMQA000006 need to be restored.

In this case, restore the receivers individually in reverse chronological order. This is not always necessary, but is good practice. In severe situations, you might need to use the i5/OS command WRKJRNA to associate the restored journal receivers with the journal.

When restoring journals, the system automatically creates an attached journal receiver with a new name in the journal receiver sequence. However, the new name generated might be the same as a journal receiver you need to restore. Manual intervention is needed to overcome this problem; to create a new name journal receiver in sequence, and new journal before restoring the journal receiver.

For instance, consider the problem with saved journal AMQAJRN and the following journal receivers:

- AMQA000000
- AMQA100000
- AMQA200000
- AMQA300000
- AMQA400000
- AMQA500000
- AMQA600000
- AMQA700000
- AMQA800000
- AMQA900000

When restoring journal AMQAJRN to a queue manager library, the system automatically creates journal receiver AMQA000000. This automatically generated receiver conflicts with one of the existing journal receivers (AMQA000000) you want to restore, which you cannot restore.

The solution is:

1. Manually create the next journal receiver (see "WebSphere MQ for i5/OS journals" on page 93):

   ```
   CRTJRNRCV JRNRCV(QMQRLIB/AMQA900001) THRESHOLD(XXXXX)
   ```

2. Manually create the journal with the above journal receiver:

   ```
   CRTJRN JRN(QMGRLIB/AMQAJRN) MNGRCV(*SYSTEM)  +
   JRNRCV(QMGRLIB/AMQA9000001) MSGQ(QMGRLIB/AMQAJRNMSG)
   ```

3. Restore the local journal receivers AMQA000000 to AMQA900000.

Performance and failover considerations

If you use a large number of persistent messages or large messages in your applications, there is an associated overhead of journaling these messages.

This increases your system disk input/output. If this disk input/output becomes excessive, performance suffers.

Ensure that you have sufficient disk activation to cope with this possibility, or consider a separate ASP in which to hold your queue manager journal receivers.

You can specify which ASP your queue manager library and journals are stored on when you create your queue manager using the ASP parameter of **CRTMQM**. By default, the queue manager library and journals and IFS data are stored in the same ASP: the system ASP.

ASPs allow isolation of objects on one or more specific disk units. This can also reduce the loss of data because of a disk media failure. In most cases, only the data that is stored on disk units in the affected ASP is lost.

You are recommended to store the queue manager library and journal data in separate user ASPs to that of the root IFS file system to provide failover and reduce disk contention.

For more information, see *i5/OS V4R4M0 Backup and Recovery*

Using SAVLIB to save WebSphere MQ libraries

You cannot use `SAVLIB LIB(*ALLUSR)` to save the WebSphere MQ libraries, because these libraries have names beginning with Q.

You can use `SAVLIB LIB(QM*)` to save all the queue manager libraries, but only if you are using a save device other than *SAVF. For `DEV(*SAVF)`, you must use a SAVLIB command for each and every queue manager library on your system.

Chapter 8. Analyzing problems

This chapter suggests reasons for problems you might have with WebSphere MQ for i5/OS, to help you in problem determination. You usually start with a symptom, or set of symptoms, and trace them back to their cause.

Do not confuse problem determination with problem solving although the process of problem determination often enables you to solve a problem. For example, if you find that the cause of the problem is an error in an application program, you can solve the problem by correcting the error.

However, you might not always be able to solve a problem after determining its cause. For example:

- A performance problem might be caused by a limitation of your hardware.
- The cause of the problem might be in the WebSphere MQ for i5/OS code. If this happens, you need to contact your IBM® support center for a solution.

This chapter is divided into the following sections:

- "Preliminary checks"
- "Problem characteristics" on page 107
- "Determining problems with WebSphere MQ applications" on page 110
- "Obtaining diagnostic information" on page 113
- "Error logs" on page 117
- "Dead-letter queues" on page 120
- "First Failure Support Technology (FFST)" on page 120
- "Performance considerations" on page 122

Preliminary checks

Before you start problem determination in detail, it is worth considering the facts to see if there is something obvious with which to start your investigation. This approach to debugging can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

The cause of your problem could be in any of the following:

- Hardware
- Operating system
- Related software, for example, a language compiler
- The network
- The WebSphere MQ product
- Your WebSphere MQ application
- Other applications
- Site operating procedures

The sections that follow raise some fundamental questions that you need to consider.

As you go through the questions, make a note of anything that might be relevant to the problem. Even if your observations do not suggest a cause immediately, they could be useful later if you have to carry out a systematic problem determination exercise.

The following steps are intended to help you isolate the problem and are taken from the viewpoint of a WebSphere MQ application. Check all the suggestions at each stage.

1. Has WebSphere MQ for i5/OS run successfully before?

   Yes Proceed to Step 2.

   No It is likely that you have not installed or set up WebSphere MQ correctly.

2. Has the WebSphere MQ application run successfully before?

   Yes Proceed to Step 3.

   No Consider the following:

   a. The application might have failed to compile or link, and fails if you attempt to invoke it. Check the output from the compiler or linker.

      Refer to the appropriate programming language reference manual, or the WebSphere MQ Application Programming Guide, for information on how to build your application.

   b. Consider the logic of the application. For example, do the symptoms of the problem indicate that a function is failing and, therefore, that a piece of code is in error.

      Check the following common programming errors:

      - Assuming that queues can be shared, when they are in fact exclusive.
      - Trying to access queues and data without the correct security authorization.
      - Passing incorrect parameters in an MQI call; if the wrong number of parameters is passed, no attempt can be made to complete the completion code and reason code fields, and the task is ended abnormally.
      - Failing to check return codes from MQI requests.
      - Using incorrect addresses.
      - Passing variables with incorrect lengths specified.
      - Passing parameters in the wrong order.
      - Failing to initialize *MsgId* and *CorrelId* correctly.

3. Has the WebSphere MQ application changed since the last successful run?

   Yes It is likely that the error lies in the new or modified part of the application. Check all the changes and see if you can find an obvious reason for the problem.

   a. Have all the functions of the application been fully exercised before?

      Could it be that the problem occurred when part of the application that had never been invoked before was used for the first time? If so, it is likely that the error lies in that part of the application. Try to find out what the application was doing when it failed, and check the source code in that part of the program for errors.

b. If the program has run successfully before, check the current queue status and files that were being processed when the error occurred. It is possible that they contain some unusual data value that causes a rarely used path in the program to be invoked.

c. The application received an unexpected MQI return code. For example:

- Does your application assume that the queues it accesses are shareable? If a queue has been redefined as exclusive, can your application deal with return codes indicating that it can no longer access that queue?
- Have any queue definition or security profiles been changed? An MQOPEN call could fail because of a security violation; can your application recover from the resulting return code?

Refer to the information in the WebSphere MQ Application Programming Reference for your programming language for a description of each return code.

d. If you have applied any PTF to WebSphere MQ for i5/OS, check that you received no error messages when you installed the PTF.

No Ensure that you have eliminated all the preceding suggestions and proceed to Step 4.

4. Has the server system remained unchanged since the last successful run?

Yes Proceed to "Problem characteristics."

No Consider all aspects of the system and review the appropriate documentation on how the change might have impacted the WebSphere MQ application. For example :

- Interfaces with other applications
- Installation of new operating system or hardware
- Application of PTFs
- Changes in operating procedures

Problem characteristics

Perhaps the preliminary checks have enabled you to find the cause of the problem. If so, you can probably now resolve it, possibly with the help of other books in the WebSphere MQ library, and in the libraries of other licensed programs.

If you have not yet found the cause, start to look at the problem in greater detail. Use the following questions as pointers to the problem. Answering the appropriate question, or questions, should lead you to the cause of the problem.

Can you reproduce the problem?

If you can reproduce the problem, consider the conditions under which you do so:

- Is it caused by a command?

  Does the operation work if it is entered by another method? If the command works if it is entered on the command line, but not otherwise, check that the command server has not stopped, and that the queue definition of the `SYSTEM.ADMIN.COMMAND.QUEUE` has not been changed.
- Is it caused by a program? If so, does it fail in batch? Does it fail on all WebSphere MQ for i5/OS systems, or only on some?

- Can you identify any application that always seems to be running in the system when the problem occurs? If so, examine the application to see if it is in error.
- Does the problem occur with any queue manager, or when connected to one specific queue manager?
- Does the problem occur with the same type of object on any queue manager, or only one particular object? What happens after this object has been cleared or redefined?
- Is the problem independent of any message persistence settings?
- Does the problem occur only when syncpoints are used?
- Does the problem occur only when one or more queue-manager events are enabled?

Is the problem intermittent?

An intermittent problem could be caused by failing to take into account the fact that processes can run independently of each other. For example, a program might issue an MQGET call, without specifying a wait option, before an earlier process has completed. You might also encounter this if your application tries to get a message from a queue while the call that put the message is in-doubt (that is, before it has been committed or backed out).

Problems with commands

Be careful when including special characters, for example back slash (\) and double quote (") characters, in descriptive text for some commands. If you use either of these characters in descriptive text, precede them with a \, that is, enter \\ or \" if you want \ or " in your text.

Queue managers and their associated object names are case sensitive. By default, i5/OS uses uppercase characters, unless you surround the name in quotes.

For example, `MYQUEUE` and `myqueue` translate to `MYQUEUE`, whereas `'myqueue'` translates to `myqueue`.

Does the problem affect all users of the WebSphere MQ for i5/OS application?

If the problem affects only some users, look for differences in how the users configure their systems and queue manager settings.

Check the library lists and user profiles. Can the problem be circumvented by having `*ALLOBJ` authority?

Does the problem affect specific parts of the network?

You might be able to identify specific parts of the network that are affected by the problem (remote queues, for example). If the link to a remote message queue manager is not working, the messages cannot flow to a remote queue.

Check the following:

- Is the connection between the two systems available, and has the intercommunication component of WebSphere MQ for i5/OS started?

Check that messages are reaching the transmission queue, the local queue definition of the transmission queue, and any remote queues.

- Have you made any network-related changes that might account for the problem or changed any WebSphere MQ for i5/OS definitions?
- Can you distinguish between a channel definition problem and a channel message problem?

  For example, redefine the channel to use an empty transmission queue. If the channel starts correctly, the definition is correctly configured.

Does the problem occur only on WebSphere MQ

If the problem occurs only on this version of WebSphere MQ, check the appropriate database on RETAIN®, or the Web site `http://www-306.ibm.com/software/integration/wmq/support/`, to ensure that you have applied all the relevant PTFs.

Does the problem occur at specific times of the day?

If the problem occurs at specific times of day, it could be that it is dependent on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, and so these are the times when load-dependent problems are most likely to occur. (If your WebSphere MQ for i5/OS network extends across more than one time zone, peak system loading might seem to occur at some other time of day.)

Have you failed to receive a response from a command?

If you have issued a command but you have not received a response, consider the following questions:

- Is the command server running?

  Work with the **DSPMQMCSVR** command to check the status of the command server.

  – If the response to this command indicates that the command server is not running, use the **STRMQMCSVR** command to start it.

  – If the response to the command indicates that the SYSTEM.ADMIN.COMMAND.QUEUE is not enabled for MQGET requests, enable the queue for MQGET requests.
- Has a reply been sent to the dead-letter queue?

  The dead-letter queue header structure contains a reason or feedback code describing the problem. See the WebSphere MQ Application Programming Reference for information about the dead-letter queue header structure (MQDLH).

  If the dead-letter queue contains messages, you can use the provided browse sample application (amqsbcg) to browse the messages using the MQGET call. The sample application steps through all the messages on a named queue for a named queue manager, displaying both the message descriptor and the message context fields for all the messages on the named queue.
- Has a message been sent to the error log?

  See “Error logs” on page 117 for further information.
- Are the queues enabled for put and get operations?
- Is the *WaitInterval* long enough?

If your MQGET call has timed out, a completion code of MQCC_FAILED and a reason code of MQRC_NO_MSG_AVAILABLE are returned. (See the WebSphere MQ Application Programming Reference for information about the *WaitInterval* field, and completion and reason codes from MQGET.)

- If you are using your own application program to put commands onto the SYSTEM.ADMIN.COMMAND.QUEUE, do you need to take a syncpoint?

  Unless you have specifically excluded your request message from syncpoint, you need to take a syncpoint before attempting to receive reply messages.
- Are the MAXDEPTH and MAXMSGL attributes of your queues set sufficiently high?
- Are you using the *CorrelId* and *MsgId* fields correctly?

  Set the values of *MsgId* and *CorrelId* in your application to ensure that you receive all messages from the queue.

Try stopping the command server and then restarting it, responding to any error messages that are produced.

If the system still does not respond, the problem could be with either a queue manager or the whole of the WebSphere MQ system. First try stopping individual queue managers to try and isolate a failing queue manager. If this does not reveal the problem, try stopping and restarting WebSphere MQ, responding to any messages that are produced in the error log.

If the problem still occurs after restart, contact your IBM Support Center for help.

If you have still not identified the cause of the problem, see "Determining problems with WebSphere MQ applications."

Determining problems with WebSphere MQ applications

This section discusses problems you might encounter with WebSphere MQ applications, commands, and messages.

Are some of your queues working?

If you suspect that the problem occurs with only a subset of queues, select the name of a local queue that you think is having problems.

1. Display the information about this queue, using WRKMQMQSTS or DSPMQMQ.
2. Use the data displayed to do the following checks:
   - If CURDEPTH is at MAXDEPTH, the queue is not being processed. Check that all applications are running normally.
   - If CURDEPTH is not at MAXDEPTH, check the following queue attributes to ensure that they are correct:
     - If triggering is being used:
       - Is the trigger monitor running?
       - Is the trigger depth too big?
       - Is the process name correct?
     - Can the queue be shared? If not, another application could already have it open for input.
     - Is the queue enabled appropriately for GET and PUT?

- If there are no application processes getting messages from the queue, determine why this is so (for example, because the applications need to be started, a connection has been disrupted, or because the MQOPEN call has failed for some reason).

If you cannot solve the problem, contact your IBM support center for help.

Does the problem affect only remote queues?

If the problem affects only remote queues, check the following:

1. Check that the programs that should be putting messages to the remote queues have run successfully.
2. If you use triggering to start the distributed queuing process, check that the transmission queue has triggering set on. Also, check that the trigger monitor is running.
3. If necessary, start the channel manually. See WebSphere MQ Intercommunication for information about how to do this.
4. Check the channel with a PING command.

See WebSphere MQ Intercommunication for information about how to define channels.

Does the problem affect messages?

This section deals with message problems, including:

- "Messages do not appear on the queue"
- "Messages contain unexpected or corrupted information" on page 112
- "Unexpected messages are received when using distributed queues" on page 113

Messages do not appear on the queue

A list of things to consider if messages do not appear on queues when you are expecting them.

If messages do not appear when you are expecting them, check for the following:

- Have you selected the correct queue manager, that is, the default queue manager or a named queue manager?
- Has the message been put on the queue successfully?
  - Has the queue been defined correctly, for example, is MAXMSGLEN sufficiently large?
  - Can applications put messages on the queue (is the queue enabled for putting)?
  - Is the queue already full? This could mean that an application was unable to put the required message on the queue.
- Can you get the message from the queue?
  - Do you need to take a syncpoint?

    If messages are being put or retrieved within syncpoint, they are not available to other tasks until the unit of recovery has been committed.
  - Is your timeout interval long enough?
  - Are you waiting for a specific message that is identified by a message identifier or correlation identifier (*MsgId* or *CorrelId*)?

Check that you are waiting for a message with the correct *MsgId* or *CorrelId*. A successful MQGET call sets both these values to that of the message retrieved, so you might need to reset these values in order to get another message successfully.

Also check if you can get other messages from the queue.

- Can other applications get messages from the queue?
- Was the message you are expecting defined as persistent?

  If not, and WebSphere MQ for i5/OS has been restarted, the message will have been lost.

If you cannot find anything wrong with the queue, and the queue manager itself is running, make the following checks on the process that you expected to put the message on to the queue:

- Did the application start?

  If it should have been triggered, check that the correct trigger options were specified.
- Is a trigger monitor running?
- Was the trigger process defined correctly?
- Did it complete correctly?

  Look for evidence of an abnormal end in the job log.
- Did the application commit its changes, or were they backed out?

If multiple transactions are serving the queue, they might occasionally conflict with one another. For example, one transaction might issue an MQGET call with a buffer length of zero to find out the length of the message, and then issue a specific MQGET call specifying the *MsgId* of that message. However, in the meantime, another transaction might have issued a successful MQGET call for that message, so the first application receives a completion code of MQRC_NO_MSG_AVAILABLE. Applications that are expected to run in a multi-server environment must be designed to cope with this situation.

Consider that the message could have been received, but that your application failed to process it in some way. For example, did an error in the expected format of the message cause your program to reject it? If this is the case, refer to "Messages contain unexpected or corrupted information."

Messages contain unexpected or corrupted information

A list of points to consider if messages are corrupted or do not contain what you expected.

If the information contained in the message is not what your application was expecting, or has been corrupted in some way, consider the following points:

- Has your application, or the application that put the message on to the queue, changed?

  Ensure that all changes are simultaneously reflected on all systems that need to be aware of the change.

  For example, a copyfile formatting the message might have been changed, in which case, re-compile both applications to pick up the changes. If one application has not been recompiled, the data appears corrupt to the other.
- Is an application sending messages to the wrong queue?

Check that the messages your application is receiving are not really intended for an application servicing a different queue. If necessary, change your security definitions to prevent unauthorized applications from putting messages on to the wrong queues.

If your application has used an alias queue, check that the alias points to the correct queue.

- Has the trigger information been specified correctly for this queue?

  Check that your application should have been started, or should a different application have been started?
- Has the CCSID been set correctly, or is the message format incorrect because of data conversion.

If these checks do not enable you to solve the problem, check your application logic, both for the program sending the message, and for the program receiving it.

Unexpected messages are received when using distributed queues

If your application uses distributed queues, consider the following points:

- Has distributed queuing been correctly installed on both the sending and receiving systems?
- Are the links available between the two systems?

  Check that both systems are available, and connected to WebSphere MQ for i5/OS. Check that the connection between the two systems is active.
- Is triggering set on in the sending system?
- Is the message you are waiting for a reply message from a remote system?

  Check that triggering is activated in the remote system.
- Is the queue already full?

  This could mean that an application was unable to put the required message on to the queue. If this is so, check if the message has been put onto the undelivered-message queue.

  The dead-letter queue message header (dead-letter header structure) contains a reason or feedback code explaining why the message could not be put on to the target queue. See the WebSphere MQ Application Programming Reference or the WebSphere MQ for i5/OS Application Programming Reference (ILE/RPG), as appropriate, for information about the dead-letter header structure.
- Is there a mismatch between the sending and receiving queue managers?

  For example, the message length could be longer than the receiving queue manager can handle.
- Are the channel definitions of the sending and receiving channels compatible?

  For example, a mismatch in sequence number wrap stops the distributed queuing component. See WebSphere MQ Intercommunication for more information about distributed queuing.

Obtaining diagnostic information

You can find diagnostic information about WebSphere MQ in the following places:

User's job log
: The job log records the commands processed by the job and the messages returned from running those commands.

Reviewing the job log of a user who experiences a problem, by issuing the DSPJOBLOG command, identifies the WebSphere MQ commands issued and the sequence of those commands.

WebSphere MQ job log
WebSphere MQ specific jobs, for example, the command server and channel programs, run under the WebSphere MQ profile QMQM. If you have a problem in these areas, review these job logs by issuing the command `WRKSPLF QMQM` to display them.

System history log
Reviewing the history log, by issuing the DSPLOG command, displays information about the operation of the system and system status. This can be useful for identifying channel connection problems.

i5/OS message queue
It is useful to view messages sent to various i5/OS message queues using the DSPMSG command. Use the command `DSPMSG QSYSOPR` to check the system operator message queue, used for WebSphere MQ journaling messages, and job completion messages in particular.

Error logs in the IFS
See "Error logs" on page 117 for further information on using the error logs generated.

Generation of FFSTs
See "First Failure Support Technology (FFST)" on page 120 for further information on First Failure Support Technology™ and an example of a WebSphere MQ for i5/OS FFST™ report.

Using WebSphere MQ for i5/OS trace

Although you need to use certain traces on occasion, running the trace facility slows your systems.

You also need to consider to what destination you want your trace information sent.

Note:

1. To run the WebSphere MQ for i5/OS trace commands, you must have the appropriate authority.
2. Trace data is written to IFS only when trace is ended, with option *OFF.
3. Trace data remains in the system until you delete it from the IFS.

Trace usage

Use the TRCMQM command to start and stop tracing and specify the type of trace that you require.

There are two stages in using trace:

1. Decide whether you want early tracing. Early tracing lets you trace the creation and startup of queue managers. Note, however, that early trace can easily generate large amounts of trace, because it is implemented by tracing **all** jobs for **all** queue managers. To enable early tracing, use TRCMQM with the TRCEARLY parameter set to *YES.
2. Start tracing WebSphere MQ work using `TRCMQM` *ON*. To stop the trace, you have two options:

- `TRCMQM` **OFF*, to stop collecting trace records for a queue manager. The trace records are written to files in the `/QIBM/UserData/mqm/trace` directory.
- `TRCMQM` **END*, to stop collecting trace records for **all** queue managers and to disable early trace. This option ignores the value of the TRCEARLY parameter.

Specify the level of detail you want, using the TRCLEVEL parameter set to one of the following values:

**DFT* For minimum-detail level for flow processing trace points.

**DETAIL*
For high-detail level for flow processing trace points.

**PARMS*
For default-detail level for flow processing trace points.

Specify the type of trace output you want, using the OUTPUT parameter set to one of the following values:

***MQM**
Collect binary WebSphere MQ trace output in the directory specified by the TRCDIR parameter. This is the default value

***MQMFMT**
Collect formatted WebSphere MQ trace output in the directory specified by the TRCDIR parameter. This is the default value

***PEX** Collect Performance Explorer (PEX) trace output

***ALL** Collect both WebSphere MQ unformatted trace and PEX trace output

Selective trace

You can reduce the amount of trace data being saved, improving run-time performance, using the command `TRCMQM` with `F4=prompt`, then `F9` to customize the TRCTYPE and EXCLUDE parameters:

TRCTYPE
Specifies the type of trace data to store in the trace file. If you omit this parameter, all trace points except those specified in EXCLUDE are enabled.

EXCLUDE
Specifies the type of trace data to **omit** from the trace file. If you omit this parameter, all trace points specified in TRCTYPE are enabled.

The options available on both TRCTYPE and EXCLUDE are:

***ALL (TRCTYPE only)**
All the trace data as specified by the following keywords is stored in the trace file.

trace-type-list
You can specify more than one option from the following keywords, but each option can appear only once.

***API** Output data for trace points associated with the MQI and major queue manager components.

***CMTRY**
Output data for trace points associated with comments in the WebSphere MQ components.

***COMMS**
Output data for trace points associated with data flowing over communications networks.

***CSDATA**
Output data for trace points associated with internal data buffers in common services.

***CSFLOW**
Output data for trace points associated with processing flow in common services.

***LQMDATA**
Output data for trace points associated with internal data buffers in the local queue manager.

***LQMFLOW**
Output data for trace points associated with processing flow in the local queue manager.

***OTHDATA**
Output data for trace points associated with internal data buffers in other components.

***OTHFLOW**
Output data for trace points associated with processing flow in other components.

***RMTDATA**
Output data for trace points associated with internal data buffers in the communications component.

***RMTFLOW**
Output data for trace points associated with processing flow in the communications component.

***SVCDATA**
Output data for trace points associated with internal data buffers in the service component.

***SVCFLOW**
Output data for trace points associated with processing flow in the service component.

***VSNDATA**
Output data for trace points associated with the version of WebSphere MQ running.

Wrapping trace

Use the MAXSTG parameter to wrap trace, and to specify the maximum size of storage to be used for the collected trace records. The options are:

DFT* Trace wrapping is **not enabled. For each job, trace data is written to a file with the suffix .TRC until tracing is stopped.

maximum-K-bytes
Trace wrapping is enabled. When the trace file reaches its maximum size, it is renamed with the suffix .TRS, and a new trace file with suffix .TRC is opened. Any existing .TRS file is deleted. Specify a value in the range 1 through 16 000.

Formatting trace output

To format any trace output:

- Enter the QShell
- Enter the command

```
/QSYS.LIB/QMQM.LIB/DSPMQTRC.PGM [-t Format] [-h] [-s]
  [-o OutputFileName] InputFileName
```

where:

InputFileName
Is a **required** parameter specifying the name of the file containing the unformatted trace. For example `/QIBM/UserData/mqm/trace/AMQ12345.TRC`.

-t *FormatTemplate*
Specifies the name of the template file containing details of how to display the trace. The default value is `/QIBM/ProdData/mqm/lib/amqtrc.fmt`.

-h Omit header information from the report.

-s Extract trace header and put to stdout.

-o *output_filename*
The name of the file into which to write formatted data.

You can also specify `dspmqtrc *` to format all trace.

Error logs

WebSphere MQ uses a number of error logs to capture messages concerning the operation of WebSphere MQ itself, any queue managers that you start, and error data coming from the channels that are in use.

The location of the error logs depends on whether the queue manager name is known.

In the IFS:

- If the queue manager name is known and the queue manager is available, error logs are located in:
  `/QIBM/UserData/mqm/qmname/errors`
- If the queue manager is not available, error logs are located in:
  `/QIBM/UserData/mqm/&SYSTEM/errors`

You can use the system utility EDTF to browse the errors directories and files. For example:

```
EDTF '/QIBM/UsedData/mqm/errors'
```

Alternatively, you can use option 23 against the queue manager from the WRKMQM panel.

Log files

At installation time, an &SYSTEM errors subdirectory is created in the IFS. The errors subdirectory can contain up to three error log files named:

- AMQERR01.LOG
- AMQERR02.LOG

- AMQERR03.LOG

After you have created a queue manager, three error log files are created when they are needed by the queue manager. These files have the same names as the &SYSTEM ones, that is AMQERR01, AMQERR02, and AMQERR03, and each has a capacity of 256 KB. The files are placed in the errors subdirectory of each queue manager that you create, that is `/QIBM/UserData/mqm/qmname/errors`.

As error messages are generated, they are placed in AMQERR01. When AMQERR01 gets bigger than 256 KB, it is copied to AMQERR02. Before the copy, AMQERR02 is copied to AMQERR03.LOG. The previous contents, if any, of AMQERR03 are discarded.

The latest error messages are thus always placed in AMQERR01, the other files being used to maintain a history of error messages.

All messages relating to channels are also placed in the appropriate queue manager's errors files, unless the name of their queue manager is unknown or the queue manager is unavailable. When the queue manager name is unavailable or its name cannot be determined, channel-related messages are placed in the &SYSTEM errors subdirectory.

To examine the contents of any error log file, use your system editor, EDTF, to view the stream files in the IFS.

Note:

1. Do ***not*** change ownership of these error logs.
2. If any error log file is deleted, it is automatically re-created when the next error message is logged.

Early errors

There are a number of special cases where the error logs have not yet been established and an error occurs. WebSphere MQ attempts to record any such errors in an error log. The location of the log depends on how much of a queue manager has been established.

If, because of a corrupt configuration file, for example, no location information can be determined, errors are logged to an errors directory that is created at installation time.

If both the WebSphere MQ configuration file and the DefaultPrefix attribute of the AllQueueManagers stanza are readable, errors are logged in the errors subdirectory of the directory identified by the DefaultPrefix attribute.

Operator messages

Operator messages identify normal errors, typically caused directly by users doing things like using parameters that are not valid on a command. Operator messages are national language enabled, with message catalogs installed in standard locations.

These messages are written to the joblog, if any. In addition, some operator messages are written to the AMQERR01.LOG file in the queue manager directory, and others to the &SYSTEM directory copy of the error log.

An example WebSphere MQ error log

Figure 13 shows a typical extract from a WebSphere MQ error log.

```
************Beginning of data**************
07/19/02  11:15:56 AMQ9411: Repository manager ended normally.

EXPLANATION:
Cause . . . . . :   The repository manager ended normally.
Recovery  . . . :   None.
Technical Description . . . . . :   None.
--------------------------------------------------------------------------------
07/19/02  11:15:57 AMQ9542: Queue manager is ending.

EXPLANATION:
Cause . . . . . :   The program will end because the queue manager is quiescing.
Recovery  . . . :   None.
Technical Description . . . . . :   None.
----- amqrimna.c : 773 ------------------------------------------------------

07/19/02  11:16:00 AMQ8004: WebSphere MQ queue manager 'mick' ended.
EXPLANATION:
Cause . . . . . :   WebSphere MQ queue manager 'mick' ended.
Recovery  . . . :   None.
Technical Description . . . . . :   None.
--------------------------------------------------------------------------------
07/19/02  11:16:48 AMQ7163: WebSphere MQ job number 18429 started.

EXPLANATION:
Cause . . . . . :   This job has started to perform work for Queue Manager
                    mick, The job's PID is 18429 the CCSID is 37. The job name is
                    582775/MQUSER/AMQZXMA0.
Recovery  . . . :   None
--------------------------------------------------------------------------------
07/19/02  11:16:49 AMQ7163: WebSphere MQ job number 18430 started.

EXPLANATION:
Cause . . . . . :   This job has started to perform work for Queue Manager
                    mick, The job's PID is 18430 the CCSID is 0. The job name is
                    582776/MQUSER/AMQZFUMA.
Recovery  . . . :   None
--------------------------------------------------------------------------------
07/19/02  11:16:49 AMQ7163: WebSphere MQ job number 18431 started.

EXPLANATION:
Cause . . . . . :   This job has started to perform work for Queue Manager
                    mick, The job's PID is 18431 the CCSID is 37. The job name is
                    582777/MQUSER/AMQZXMAX.
Recovery  . . . :   None
--------------------------------------------------------------------------------
07/19/02  11:16:50 AMQ7163: WebSphere MQ job number 18432 started.

EXPLANATION:
Cause . . . . . :   This job has started to perform work for Queue Manager
                    mick, The job's PID is 18432 the CCSID is 37. The job name is
                    582778/MQUSER/AMQALMPX.
Recovery  . . . :   None
--------------------------------------------------------------------------------
```

Figure 13. Extract from a WebSphere MQ error log

Dead-letter queues

Messages that cannot be delivered for some reason are placed on the dead-letter queue. You can check whether the queue contains any messages by issuing an MQSC DISPLAY QUEUE command. If the queue contains messages, you can use the provided browse sample application (amqsbcg) to browse messages on the queue using the MQGET call. The sample application steps through all the messages on a named queue for a named queue manager, displaying both the message descriptor and the message context fields for all the messages on the named queue.

You must decide how to dispose of any messages found on the dead-letter queue, depending on the reasons for the messages being put on the queue.

Problems might occur if you do not associate a dead-letter queue with each queue manager. For more information about dead-letter queues, see Chapter 6, "The WebSphere MQ dead-letter queue handler," on page 83.

First Failure Support Technology (FFST)

Describes the role of First Failure Support Technology (FFST).

For i5/OS, FFST information is recorded in a stream file in the `/QIBM/UserData/mqm/errors` directory.

These errors are normally severe, unrecoverable errors, and indicate either a configuration problem with the system or a WebSphere MQ internal error.

The stream files are named `AMQ`*`nnnnn`*`.`*`mm`*`.FDC`, where:

nnnnn	Is the ID of the process reporting the error
mm	Is a sequence number, normally 0

A copy of the failing job's job log is written to a file with the same name as the .FDC file. The file name ends with .JOB.

Some typical FFST data is shown in the following example.

```
------------------------------------------------------------------------------
| WebSphere MQ First Failure Symptom Report                                  |
| =====================================                                      |
|                                                                            |
| Date/Time         :- Mon January 28 2008 21:59:06 GMT                      |
| UTC Time/Zone     :- 1201539869.892015 0 GMT                               |
| Host Name         :- WINAS12B.HURSLEY.IBM.COM                              |
| PIDS              :- 5733A38                                               |
| LVLS              :- 520                                                   |
| Product Long Name :- WebSphere MQ for i5/OS                                |
| Vendor            :- IBM                                                   |
| Probe Id          :- XY353001                                              |
| Application Name  :- MQM                                                   |
| Component         :- xehAS400ConditionHandler                              |
| Build Date        :- Feb 25 2008                                           |
| UserID            :- 00000331 (MAYFCT)                                     |
| Program Name      :- STRMQM_R  MAYFCT                                      |
| Job Name          :- 020100/MAYFCT/STRMQM_R                                |
| Activation Group  :- 101 (QMQM) (QMQM/STRMQM_R)                            |
| Process           :- 00001689                                              |
| Thread            :- 00000001                                              |
| QueueManager      :- TEST.AS400.OE.P                                       |
| Major Errorcode   :- STOP                                                  |
| Minor Errorcode   :- OK                                                    |
```

```
| Probe Type        :- HALT6109                                                 |
| Probe Severity    :- 1                                                        |
| Probe Description :- 0                                                        |
| Arith1            :- 1 1                                                      |
| Comment1          :-    00d0                                                  |
--------------------------------------------------------------------------------

MQM Function Stack
lpiSPIMQConnect
zstMQConnect
ziiMQCONN
ziiClearUpAgent
xcsTerminate
xlsThreadInitialization
xcsConnectSharedMem
xstConnSetInSPbyHandle
xstConnSharedMemSet
xcsFFST

MQM Trace History
              <-- xcsCheckProcess rc=xecP_E_INVALID_PID
              --> xcsCheckProcess
              <-- xcsCheckProcess rc=xecP_E_INVALID_PID
              --> xlsThreadInitialization
               --> xcsConnectSharedMem
                --> xcsRequestThreadMutexSem
                <-- xcsRequestThreadMutexSem rc=OK
                --> xihGetConnSPDetailsFromList
                <-- xihGetConnSPDetailsFromList rc=OK
                --> xstCreateConnExtentList
                <-- xstCreateConnExtentList rc=OK
                --> xstConnSetInSPbyHandle
                 --> xstSerialiseSPList
                  --> xllSpinLockRequest
                  <-- xllSpinLockRequest rc=OK
                 <-- xstSerialiseSPList rc=OK
                 --> xstGetSetDetailsFromSPByHandle
                 <-- xstGetSetDetailsFromSPByHandle rc=OK
                 --> xstConnSharedMemSet
                  --> xstConnectExtent
                   --> xstAddConnExtentToList
                   <-- xstAddConnExtentToList rc=OK
                  <-- xstConnectExtent rc=OK
                  --> xcsBuildDumpPtr
                   --> xcsGetMem
                   <-- xcsGetMem rc=OK
                  <-- xcsBuildDumpPtr rc=OK
                  --> xcsBuildDumpPtr
                  <-- xcsBuildDumpPtr rc=OK
                  --> xcsBuildDumpPtr
                  <-- xcsBuildDumpPtr rc=OK
                  --> xcsFFST

Process Control Block
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :8bba0:0:6d   E7C9C8D7  000004E0  00000699  00000000    XIHP...\...r....
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :8bbb0:1:6d   00000000  00000002  00000000  00000000    ................
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :8bbc0:2:6d   80000000  00000000  EC161F7C  FC002DB0    ...........@...¢
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :8bbd0:3:6d   80000000  00000000  EC161F7C  FC002DB0    ...........@...¢
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :8bbe0:4:6d   00000000  00000000  00000000  00000000    ................

Thread Control Block
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :1db0:20:6d   E7C9C8E3  00001320  00000000  00000000    XIHT............
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :1dc0:21:6d   00000001  00000000  00000000  00000000    ................
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :1dd0:22:6d   80000000  00000000  DD13C17B  81001000    ..........A#a...
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :1de0:23:6d   00000000  00000046  00000002  00000001    ................
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :1df0:24:6d   00000000  00000000  00000000  00000000    ................


RecoveryIndex
SPP:0000 :1aefSTRMQM_R  MAYFCT    020100 :2064:128:6d   00000000                                   ....
```

Note:

1. The `MQM Trace History` section is a log of the 200 most recent function trace statements, and is recorded in the FFST report regardless of any TRCMQM settings.

2. The queue manager details are recorded only for jobs that are connected to a queue manager subpool.
3. When the failing component is `xehAS400ConditionHandler`, additional data is logged in the errors directory giving extracts from the joblog relating to the exception condition.

The function stack and trace history are used by IBM to assist in problem determination. In most cases there is little that the system administrator can do when an FFST report is generated, apart from raising problems through the IBM Support Center.

Performance considerations

This section discusses:

- General design considerations; see "Application design considerations"
- Specific performance problems; see "Specific performance problems" on page 123

Application design considerations

There are a number of ways in which poor program design can affect performance. These can be difficult to detect because the program can appear to perform well, while impacting the performance of other tasks. Several problems specific to programs making WebSphere MQ for i5/OS calls are discussed in the following sections.

For more information about application design, see the WebSphere MQ Application Programming Guide.

Effect of message length

Although WebSphere MQ for i5/OS allows messages to hold up to 100 MB of data, the amount of data in a message affects the performance of the application that processes the message. To achieve the best performance from your application, send only the essential data in a message; for example, in a request to debit a bank account, the only information that might need to be passed from the client to the server application is the account number and the amount of the debit.

Effect of message persistence

Persistent messages are journaled. Journaling messages reduces the performance of your application, so use persistent messages for essential data only. If the data in a message can be discarded if the queue manager stops or fails, use a nonpersistent message.

Searching for a particular message

The MQGET call usually retrieves the first message from a queue. If you use the message and correlation identifiers (*MsgId* and *CorrelId*) in the message descriptor to specify a particular message, the queue manager has to search the queue until it finds that message. The use of the MQGET call in this way affects the performance of your application.

Queues that contain messages of different lengths

If the messages on a queue are of different lengths, to determine the size of a message, your application could use the MQGET call with the *BufferLength* field set to zero so that, even though the call fails, it returns the size of the message data. The application could then repeat the call, specifying the identifier of the message it measured in its first call and a buffer of the correct size. However, if there are other applications serving the same queue, you might find that the performance of your application is reduced because its second MQGET call spends time searching for a message that another application has retrieved in the time between your two calls.

If your application cannot use messages of a fixed length, another solution to this problem is to use the MQINQ call to find the maximum size of messages that the queue can accept, then use this value in your MQGET call. The maximum size of messages for a queue is stored in the *MaxMsgLen* attribute of the queue. This method could use large amounts of storage, however, because the value of this queue attribute could be the maximum allowed by WebSphere MQ for i5/OS, which could be greater than 2 GB.

Frequency of syncpoints

Programs that issue numerous MQPUT calls within syncpoint, without committing them, can cause performance problems. Affected queues can fill up with messages that are currently unusable, while other tasks might be waiting to get these messages. This has implications in terms of storage, and in terms of threads tied up with tasks that are attempting to get messages.

Use of the MQPUT1 call

Use the MQPUT1 call only if you have a single message to put on a queue. If you want to put more than one message, use the MQOPEN call, followed by a series of MQPUT calls and a single MQCLOSE call.

Number of threads in use

An application might require a large number of threads. Each queue manager process is allocated a maximum allowable number of threads.

If some applications are troublesome, it could be due to their design using too many threads. Consider whether the application takes into account this possibility and that it takes actions either to stop or to report this type of occurrence.

The maximum number of threads that i5/OS allows is 4095. However, the default is 64. WebSphere MQ makes available up to 63 threads to its processes.

Specific performance problems

This section discusses the problems of storage and poor performance.

Storage problems

If you receive the system message `CPF0907. Serious storage condition may exist` it is possible that you are filling up the space associated with the WebSphere MQ for i5/OS queue managers.

Is your application or WebSphere MQ for i5/OS running slowly?

If your application is running slowly, this could indicate that it is in a loop, or waiting for a resource that is not available.

This could also be caused by a performance problem. Perhaps it is because your system is operating near the limits of its capacity. This type of problem is probably worst at peak system load times, typically at mid-morning and mid-afternoon. (If your network extends across more than one time zone, peak system load might seem to you to occur at some other time.)

If you find that performance degradation is not dependent on system loading, but happens sometimes when the system is lightly loaded, a poorly designed application program is probably to blame. This could manifest itself as a problem that only occurs when certain queues are accessed.

QTOTJOB and QADLTOTJ are system values worth investigating.

The following symptoms might indicate that WebSphere MQ for i5/OS is running slowly:

- If your system is slow to respond to MQSC commands.
- If repeated displays of the queue depth indicate that the queue is being processed slowly for an application with which you would expect a large amount of queue activity.
- Is MQ trace running?

Chapter 9. Configuring WebSphere MQ

This chapter explains how to change the behavior of queue managers to suit your installation's needs.

You change WebSphere MQ configuration information by modifying the values specified on a set of configuration attributes (or parameters) that govern WebSphere MQ. You change these attributes by editing the **WebSphere MQ configuration files**.

This chapter:

- Describes the methods for configuring WebSphere MQ in "WebSphere MQ configuration files."
- Describes the attributes you can use to modify configuration information in "Attributes for changing WebSphere MQ configuration information" on page 127.
- Describes the attributes you can use to modify queue manager configuration information in "Changing queue manager configuration information" on page 130.
- Provides examples of mqs.ini and qm.ini files for WebSphere MQ for i5/OS in "Example mqs.ini and qm.ini files" on page 138.

WebSphere MQ configuration files

You modify WebSphere MQ configuration attributes within:

- A WebSphere MQ configuration file (**mqs.ini**) to effect changes on the node as a whole. There is one mqs.ini file for each WebSphere MQ installation.
- A queue manager configuration file (**qm.ini**) to effect changes for specific queue managers. There is one qm.ini file for each queue manager on the node.

Note that .ini files are stream files resident in the IFS.

A configuration file (which can be referred to as a ***stanza*** file) contains one or more stanzas, which are simply groups of lines in the .ini file that together have a common function or define part of a system, for example, log functions and channel functions.

Any changes you make to a configuration file do not take effect until the next time the queue manager is started.

Editing configuration files

Before editing a configuration file, back it up so that you have a copy you can revert to if the need arises.

You can edit configuration files either:

- Automatically, using commands that change the configuration of queue managers on the node
- Manually, using the EDTF CL editor

You can edit the default values in the WebSphere MQ configuration files after installation.

If you set an incorrect value on a configuration file attribute, the value is ignored and an operator message is issued to indicate the problem. (The effect is the same as missing out the attribute entirely.)

When you create a new queue manager:

- Back up the WebSphere MQ configuration file
- Back up the new queue manager configuration file

When do you need to edit a configuration file?

You might need to edit a configuration file if, for example:

- You lose a configuration file; recover from backup if possible.
- You need to move one or more queue managers to a new directory.
- You need to change your default queue manager; this could happen if you accidentally delete the existing queue manager.
- You are advised to do so by your IBM Support Center.

Configuration file priorities

The attribute values of a configuration file are set according to the following priorities:

- Parameters entered on the command line take precedence over values defined in the configuration files.
- Values defined in the qm.ini files take precedence over values defined in the mqs.ini file.

The WebSphere MQ configuration file mqs.ini

The WebSphere MQ configuration file, mqs.ini, contains information relevant to all the queue managers on a WebSphere MQ installation. It is created automatically during installation. In particular, the mqs.ini file is used to locate the data associated with each queue manager.

The mqs.ini file is stored in `/QIBM/UserData/mqm`

The mqs.ini file contains:

- The names of the queue managers
- The name of the default queue manager
- The location of the files associated with each queue manager
- Information identifying any API exits (see “API exits” on page 134 for more information)

Queue manager configuration files qm.ini

A queue manager configuration file, qm.ini, contains information relevant to a specific queue manager. There is one queue manager configuration file for each queue manager. The qm.ini file is automatically created when the queue manager with which it is associated is created.

A qm.ini file is held in the `<mqmdata directory>/QMNAME/qm.ini`, where:

- <mqmdata directory> is /QIBM/UserData/mqm by default.
- QMNAME is the name of the queue manager to which the initialization file applies.

Note:

1. You can change the <mqmdata directory> in the mqs.ini file.
2. The queue manager name can be up to 48 characters in length. However, this does not guarantee that the name is valid or unique. Therefore, a directory name is generated based on the queue manager name. This process is known as **name transformation**. See "Understanding WebSphere MQ queue manager library names" on page 681 for further information.

Attributes for changing WebSphere MQ configuration information

The following groups of attributes appear in mqs.ini:

- "The AllQueueManagers stanza"
- "The DefaultQueueManager stanza" on page 128
- "The ExitProperties stanza" on page 128
- "The QueueManager stanza" on page 129

There are also two stanzas associated with API exits, ApiExitCommon and ApiExitTemplate. For details on using these, see "Configuring API exits" on page 136.

Note: In the descriptions of the stanzas, the value underlined is the default value and the | symbol means *or*.

The AllQueueManagers stanza

The AllQueueManagers stanza can specify:

- The path to the qmgrs directory where the files associated with a queue manager are stored
- The path to the executable library
- The method for converting EBCDIC-format data to ASCII format

DefaultPrefix=*directory_name*

The path to the qmgrs directory, below which the queue manager data is kept.

If you change the default prefix for the queue manager, you must replicate the directory structure that was created at installation time.

In particular, you must create the qmgrs structure. Stop WebSphere MQ before changing the default prefix, and restart WebSphere MQ only after moving the structures to the new location and changing the default prefix.

As an alternative to changing the default prefix, you can use the environment variable MQSPREFIX to override the DefaultPrefix for the **CRTMQM** command.

ConvEBCDICNewline=NL_TO_LF|TABLE|ISO

EBCDIC code pages contain a new line (NL) character that is not supported by ASCII code pages, although some ISO variants of ASCII contain an equivalent.

Use the ConvEBCDICNewline attribute to specify the method WebSphere MQ is to use when converting the EBCDIC NL character into ASCII format.

NL_TO_LF
Convert the EBCDIC NL character (X'15') to the ASCII line feed character, LF (X'0A'), for all EBCDIC to ASCII conversions.

NL_TO_LF is the default.

TABLE
Convert the EBCDIC NL character according to the conversion tables used on i5/OS for all EBCDIC to ASCII conversions.

Note that the effect of this type of conversion can vary from language to language .

ISO
Specify ISO if you want:

- ISO CCSIDs to be converted using the TABLE method
- All other CCSIDs to be converted using the NL_TO_CF method.

Possible ISO CCSIDs are shown in Table 12.

Table 12. List of possible ISO CCSIDs

CCSID	Code Set
819	ISO8859-1
912	ISO8859-2
915	ISO8859-5
1089	ISO8859-6
813	ISO8859-7
916	ISO8859-8
920	ISO8859-9
1051	roman8

If the ASCII CCSID is not an ISO subset, ConvEBCDICNewline defaults to NL_TO_LF.

The DefaultQueueManager stanza

The `DefaultQueueManager` stanza specifies the default queue manager for the node.

Name=*default_queue_manager*
The default queue manager processes any commands for which a queue manager name is not explicitly specified. The `DefaultQueueManager` attribute is automatically updated if you create a new default queue manager. If you inadvertently create a new default queue manager and then want to revert to the original, you must alter the `DefaultQueueManager` attribute manually.

The ExitProperties stanza

The `ExitProperties` stanza specifies configuration options used by queue manager exit programs.

CLWLMode=SAFE|FAST
The cluster workload exit, CLWL, allows you to specify which cluster queue in the cluster is to be opened in response to an MQI call (MQOPEN, MQPUT, and so on). The CLWL exit runs either in FAST mode or SAFE mode depending on

the value you specify on the CLWLMode attribute. If you omit the CLWLMode attribute, the cluster workload exit runs in SAFE mode.

SAFE
Run the CLWL exit in a separate process to the queue manager. This is the default.

If a problem arises with the user-written CLWL exit when running in SAFE mode, the following happens:

- The CLWL server process (amqzlwa0) fails
- The queue manager restarts the CLWL server process
- The error is reported to you in the error log. If an MQI call is in progress, you receive notification in the form of a bad return code.

The integrity of the queue manager is preserved.

Note: There is a possible performance overhead associated with running the CLWL exit in a separate process.

FAST
Run the cluster exit inline in the queue manager process.

Specifying this option improves performance by avoiding the overheads associated with running in SAFE mode, but does so at the expense of queue manager integrity. Run the CLWL exit in FAST mode only if you are convinced that there are **no** problems with your CLWL exit, and you are particularly concerned about performance overheads.

If a problem arises when the CLWL exit is running in FAST mode, the queue manager fails and you run the risk of compromising the integrity of the queue manager.

The QueueManager stanza

There is one `QueueManager` stanza for every queue manager. These attributes specify the queue manager name and the name of the directory containing the files associated with that queue manager. The name of the directory is based on the queue manager name, but is transformed if the queue manager name is not a valid file name.

See "Understanding WebSphere MQ queue manager library names" on page 681 for more information about name transformation.

Name=*queue_manager_name*
The name of the queue manager.

Prefix=*prefix*
Where the queue manager files are stored. By default, this is the same as the value specified on the DefaultPrefix attribute of the `AllQueueManager` stanza in the mqs.ini file.

Directory=*name*
The name of the subdirectory under the `<prefix>\QMGRS` directory where the queue manager files are stored. This name is based on the queue manager name, but can be transformed if there is a duplicate name, or if the queue manager name is not a valid file name.

Library=*name*
The name of the library where i5/OS objects pertinent to this queue manager, for example, journals and journal receivers, are stored. This name is based on

the queue manager name, but can be transformed if there is a duplicate name, or if the queue manager name is not a valid library name.

Changing queue manager configuration information

The following groups of attributes can appear in a qm.ini file particular to a given queue manager, or used to override values set in mqs.ini.

- "The Log stanza"
- "The Channels stanza"
- "The TCP stanza" on page 133

There is also a stanza associated with API exits, ApiExitLocal. For details on using this, see "Configuring API exits" on page 136.

The Log stanza

Parameters for configuring the log file.

The `Log` stanza specifies the log attributes for a particular queue manager. By default, these are inherited from the settings specified in the `LogDefaults` stanza in the mqs.ini file when the queue manager is created.

Only change attributes of this stanza if you want to configure a queue manager differently from others.

The values specified on the attributes in the qm.ini file are read when the queue manager is started. The file is created when the queue manager is created.

LogBufferSize
The journal buffer size, in bytes. Enter a number between 32 000 and 15 761 440. The default is 32 000.

LogPath=*library_name*
The name of the library used to store journals and journal receivers for this queue manager.

LogReceiverSize
The journal receiver size, in kilobytes. The default is 100 000.

The Channels stanza

The `Channels` stanza contains information about the channels.

MaxChannels=100 | *number*
The maximum number of channels allowed. The default is 100.

MaxActiveChannels=*MaxChannels_value*
The maximum number of channels allowed to be active at any time. The default is the value specified on the `MaxChannels` attribute.

MaxInitiators=3 | *number*
The maximum number of initiators. The default and maximum value is 3. Any value greater than 3 will be taken as 3.

MQIBINDTYPE=FASTPATH | STANDARD
The binding for applications.

FASTPATH
Channels connect using MQCONNX FASTPATH. That is, there is no agent process.

STANDARD
Channels connect using STANDARD.

ThreadedListener=NO | YES
Whether to start RUNMQLSR (**YES**) or AMQCLMAA (**NO**) as a listener.

If you specify ThreadedListener=YES, all channels will run as threads of a single job. This limits the number of connections to the resources available to a single job.

If you specify ThreadedListener=NO, the non-threaded listener (AMQCLMAA) starts a new responder job (AMQCRSTA) for each inbound TCP/IP channel. The disadvantage of this technique is that it is not as fast to start a new AMQCRSTA job as it is to start a thread within a RUNMQLSR job, therefore connection times for a non-threaded listener are generally slower than those for a threaded listener.

AdoptNewMCA=NO | SVR | SNDR | RCVR | CLUSRCVR | ALL | FASTPATH
If WebSphere MQ receives a request to start a channel, but finds that an amqcrsta process already exists for the same channel, the existing process must be stopped before the new one can start. The `AdoptNewMCA` attribute allows you to control the ending of an existing process and the startup of a new one for a specified channel type.

If you specify the `AdoptNewMCA` attribute for a given channel type, but the new channel fails to start because the channel is already running:

1. The new channel tries to end the previous one.
2. If the previous channel server does not end by the time the AdoptNewMCATimeout wait interval expires, the process (or the thread) for the previous channel server is ended.
3. If the previous channel server has not ended after step 2, and after the AdoptNewMCATimeout wait interval expires for a second time, WebSphere MQ ends the channel with a `CHANNEL IN USE` error.

You specify one or more values, separated by commas or blanks, from the following list:

NO
The `AdoptNewMCA` feature is not required. This is the default.

SVR
Adopt server channels

SNDR
Adopt sender channels

RCVR
Adopt receiver channels

CLUSRCVR
Adopt cluster receiver channels

ALL
Adopt all channel types, except for FASTPATH channels

FASTPATH
Adopt the channel if it is a FASTPATH channel. This happens only if the appropriate channel type is also specified, for example, AdoptNewMCA=RCVR,SVR,FASTPATH

Attention!: The AdoptNewMCA attribute can behave in an unpredictable fashion with FASTPATH channels because of the internal design of the queue manager. Exercise great caution when enabling the AdoptNewMCA attribute for FASTPATH channels.

AdoptNewMCATimeout=60|1—3600
The amount of time, in seconds, that the new process waits for the old process to end. Specify a value, in seconds, in the range 1 to 3600. The default value is 60.

AdoptNewMCACheck=QM|ADDRESS|NAME|ALL
The AdoptNewMCACheck attribute allows you to specify the type checking required when enabling the AdoptNewMCA attribute. It is important for you to perform all three of the following checks, if possible, to protect your channels from being shut down, inadvertently or maliciously. At the very least check that the channel names match.

Specify one or more values, separated by commas or blanks, from the following:

QM
The listener process checks that the queue manager names match.

ADDRESS
The listener process checks the communications address, for example, the TCP/IP address.

NAME
The listener process checks that the channel names match.

ALL
The listener process checks for matching queue manager names, the communications address, and for matching channel names.

The default is AdoptNewMCACheck=NAME,ADDRESS,QM.

The queue manager error log stanza

Use the QMErrorLog stanza in the qm.ini file to tailor the operation and contents of queue manager error logs.

ErrorLogSize=*maxsize*
Specifies the size of the queue manager error log at which it is copied to the backup. *maxsize* must be between 1048576 and 2147483648 bytes. If **ErrorLogSize** is not specified, the default value of 262144 bytes (256 KB) is used.

ExcludeMessage=*msgIds*
Specifies messages that are not to be written to the queue manager error log. *msqIds* contain a comma separated list of message IDs from the following:

7163 - Job started message (i5/OS only)
7234 - Number of messages loaded
9001 - Channel program ended normally
9002 - Channel program started
9202 - Remote host not available
9208 - Error on receive from host
9209 - Connection closed

9228 - Cannot start channel responder
9508 - Cannot connect to queue manager
9524 - Remote queue manager unavailable
9528 - User requested closure of channel
9558 - Remote Channel is not available
9999 - Channel program ended abnormally

SuppressMessage=*msgIds*
Specifies messages that will be written to the queue manager error log once only in a specified time interval. The time interval is specified by **SuppressInterval**. *msqIds* contain a comma separated list of message IDs from the following:

7163 - Job started message (i5/OS only)
7234 - Number of messages loaded
9001 - Channel program ended normally
9002 - Channel program started
9202 - Remote host not available
9208 - Error on receive from host
9209 - Connection closed
9228 - Cannot start channel responder
9508 - Cannot connect to queue manager
9524 - Remote queue manager unavailable
9528 - User requested closure of channel
9558 - Remote Channel is not available
9999 - Channel program ended abnormally

If the same message id is specified in both **SuppressMessage** and **ExcludeMessage**, the message is excluded.

SuppressInterval=*length*
Specifies the time interval, in seconds, in which messages specified in **SuppressMessage** will be written to the queue manager error log once only. *length* must be between 1 and 86400 seconds. If **SuppressInterval** is not specified, the default value of 30 seconds is used.

The TCP stanza

Use these queue manager properties pages, or stanzas in the qm.ini file, to specify network protocol configuration parameters. They override the default attributes for channels.

Note: Only attributes representing changes to the default values need to be specified.

TCP
The following attributes can be specified:

Port=1414 | *port_number*
The default port number, in decimal notation, for TCP/IP sessions. The default port number for WebSphere MQ is 1414.

KeepAlive=NO | **YES**
Switch the KeepAlive function on or off. KeepAlive=YES causes TCP/IP to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

ListenerBacklog=number
When receiving on TCP/IP, a maximum number of outstanding connection requests is set. This can be considered to be a *backlog* of requests waiting on the TCP/IP port for the listener to accept the request. The default listener backlog value for i5/OS is 255; the maximum is 512. If the backlog reaches the value of 512, the TCP/IP connection is rejected and the channel cannot start.

For MCA channels, this results in the channel going into a RETRY state and retrying the connection at a later time.

For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and should retry the connection at a later time.

The `ListenerBacklog` attribute allows you to override the default number of outstanding requests for the TCP/IP listener.

Connect_Timeout=number|0
The number of seconds before an attempt to connect the socket times out. The default value of zero specifies that there is no connect timeout.

The following group of properties can be used to control the size of buffers used by TCP/IP. The values are passed directly to the TCP/IP layer of the operating system. Great care should be taken when using these properties. If the values are set incorrectly it can adversely affect the TCP/IP performance. For further information about how this affects performance refer to the TCP/IP documentation for your environment.

SvrSndBuffSize=number|32768
The size in bytes of the TCP/IP send buffer used by the server end of the client-connection server-connection channels.

SvrRcvBuffSize=number|32768
The size in bytes of the TCP/IP receive buffer used by the server end of the client-connection server-connection channels.

API exits

API exits let you write code that changes the behavior of WebSphere MQ API calls, such as MQPUT and MQGET, and then insert that code immediately before or immediately after those calls. The insertion is automatic; the queue manager drives the exit code at the registered points.

This chapter explains why you might want to use API exits, then describes what administration tasks are involved in enabling them. The sections are:

- "Why use API exits" on page 135
- "How you use API exits" on page 135
- "What happens when an API exit runs?" on page 136
- "Configuring API exits" on page 136

We give a brief introduction to writing API exits in "How to write an API exit" on page 135. For detailed information about writing API exits, aimed at application programmers, see the WebSphere MQ Application Programming Guide.

For detailed reference information about API exits, see the WebSphere MQ System Administration Guide.

Why use API exits

There are many reasons why you might want to insert code that modifies the behavior of applications at the level of the queue manager. Each of your applications has a specific job to do, and its code should do that task as efficiently as possible. At a higher level, you might want to apply standards or business processes to a particular queue manager for **all** the applications that use that queue manager. It is more efficient to do this above the level of individual applications, and thus without having to change the code of each application affected.

Here are a few suggestions of areas in which API exits might be useful:

- For *security*, you can provide authentication, checking that applications are authorized to access a queue or queue manager. You can also police applications' use of the API, authenticating the individual API calls, or even the parameters they use.
- For *flexibility*, you can respond to rapid changes in your business environment without changing the applications that rely on the data in that environment. You could, for example, have API exits that respond to changes in interest rates, currency exchange rates, or the price of components in a manufacturing environment.
- For *monitoring* use of a queue or queue manager, you can trace the flow of applications and messages, log errors in the API calls, set up audit trails for accounting purposes, or collect usage statistics for planning purposes.

How you use API exits

This section gives a brief overview of the tasks involved in setting up API exits.

How to configure WebSphere MQ for API exits

You configure WebSphere MQ to enable API exits by editing the configuration files, mqs.ini and qm.ini, and adding new stanzas that.

- Name the API exit
- Identify the module and entry point of the API exit code to run
- Optionally pass data with the exit
- Identify the sequence of this exit in relation to other exits

For detailed information on this configuration, see "Configuring API exits" on page 136. For a description of how API exits run, see "What happens when an API exit runs?" on page 136.

How to write an API exit

This section introduces writing API exits. For detailed information, aimed at application programmers, see the WebSphere MQ Application Programming Guide.

You write your exits using the C programming language. To help you do so, we provide a sample exit, amqsaxe0, that generates trace entries to a named file. When you start writing exits, we recommend that you use this as your starting point.

Exits are available for every API call, as follows. Within API exits, these calls take the general form:

```
MQ_call_EXIT (parameters, context, ApiCallParameters)
```

where `call` is the API call name without the MQ suffix (PUT, GET, and so on), `parameters` control the function of the exit, `context` describes the context in which the API exit was called, and `ApiCallParameters` represent the parameters to the MQI call,

What happens when an API exit runs?

The API exit routines to run are identified in stanzas in mqs.ini and qm.ini. The definition of the routines can occur in three places:

1. ApiExitCommon, in the mqs.ini file, identifies routines, for the whole of WebSphere MQ, applied when queue managers start up. These can be overridden by routines defined for individual queue managers.
2. ApiExitTemplate, in the mqs.ini file, identifies routines, for the whole of WebSphere MQ, copied to the ApiExitLocal set when a new queue manager is created.
3. ApiExitLocal, in the qm.ini file, identifies routines applicable to a particular queue manager.

When a new queue manager is created, the ApiExitTemplate definitions in mqs.ini are copied to the ApiExitLocal definitions in qm.ini for the new queue manager. When a queue manager is started, both the ApiExitCommon and ApiExitLocal definitions are used. The ApiExitLocal definitions replace the ApiExitCommon definitions if both identify a routine of the same name. The `Sequence` attribute, described in "Attributes for all stanzas" determines the order in which the routines defined in the stanzas run.

Configuring API exits

This section tells you how to configure API exits.

You define your API exits in stanzas in the mqs.ini and qm.ini files. The sections below describe these stanzas, and the attributes within them that define the exit routines and the sequence in which they run. For guidance on the process of changing these stanzas, see "Changing the configuration information" on page 138.

Stanzas in mqs.ini are:

ApiExitCommon
When any queue manager starts, the attributes in this stanza are read, and then overridden by the API exits defined in qm.ini.

ApiExitTemplate
When any queue manager is created, the attributes in this stanza are copied into the newly created qm.ini file under the ApiExitLocal stanza.

The stanza in qm.ini is:

ApiExitLocal
When the queue manager starts, API exits defined here override the defaults defined in mqs.ini.

Attributes for all stanzas

All these stanzas have the following attributes:

Name=ApiExit_name
The descriptive name of the API exit passed to it in the ExitInfoName field of the MQAXP structure.

This name must be unique, no longer than 48 characters, and contain only valid characters for the names of WebSphere MQ objects (for example, queue names).

Function=function_name
The name of the function entry point into the module containing the API exit code. This entry point is the MQ_INIT_EXIT function.

The length of this field is limited to MQ_EXIT_NAME_LENGTH.

Module=module_name
The module containing the API exit code.

If this field contains the full path name of the module it is used as is.

If this field contains just the module name, the module is located using the `ExitsDefaultPath` attribute in the `ExitPath` in qm.ini.

The length of this field is limited to the maximum path length the platform supports.

Data=data_name
Data to be passed to the API exit in the ExitData field of the MQAXP structure.

If you include this attribute, leading and trailing blanks are removed, the remaining string is truncated to 32 characters, and the result is passed to the exit. If you omit this attribute, the default value of 32 blanks is passed to the exit.

The maximum length of this field is 32 characters.

Sequence=sequence_number
The sequence in which this API exit is called relative to other API exits. An exit with a **low** sequence number is called before an exit with a **higher** sequence number. There is no need for the sequence numbering of exits to be contiguous; a sequence of 1, 2, 3 has the same result as a sequence of 7, 42, 1096. If two exits have the same sequence number, the queue manager decides which one to call first. You can tell which was called *after* the event by putting the time or a marker in ExitChainArea indicated by the ExitChainAreaPtr in MQAXP or by writing your own log file.

This attribute is an unsigned numeric value.

Sample stanzas

Once you have created an exit as a service program, you must refer to it in the appropriate stanza in a configuration file using LIBRARY/PROGRAM syntax, as shown in the examples.

The mqs.ini file below contains the following stanzas:

ApiExitTemplate
This stanza defines an exit with the descriptive name `OurPayrollQueueAuditor`, module name `MYAUDIT`, and sequence number 2. A data value of 123 is passed to the exit.

ApiExitCommon
This stanza defines an exit with the descriptive name MQPoliceman, module name MYSECURE, and sequence number 1. The data passed is an instruction (CheckEverything).

mqs.ini

```
   ApiExitTemplate:
      Name=OurPayrollQueueAuditor
      Sequence=2
      Function=EntryPoint
      Module=MYLIB/MYAUDIT
      Data=123
   ApiExitCommon:
      Name=MQPoliceman
      Sequence=1
      Function=EntryPoint
      Module=MYLIB/MYSECURE
      Data=CheckEverything
```

The qm.ini file below contains an ApiExitLocal definition of an exit with the descriptive name ClientApplicationAPIchecker, module name MYCHECK, and sequence number 3.

qm.ini

```
   ApiExitLocal:
      Name=ClientApplicationAPIchecker
      Sequence=3
      Function=EntryPoint
      Module=MYLIB/MYCHECK
      Data=9.20.176.20
```

Changing the configuration information

The WebSphere MQ configuration file, mqs.ini, contains information relevant to all the queue managers on a particular node.

A queue manager configuration file, qm.ini, contains information relevant to a specific queue manager. There is one queue manager configuration file for each queue manager, stored in its own subdirectory.

The configuration files are stored in the IFS, as follows:

```
/QIBM/UserData/mqm/mqs.ini
/QIBM/UserData/mqm/qmgrs/<queue-manager-name>/qm.ini
```

Before editing a configuration file, back it up so that you have a copy you can revert to if the need arises.

You can edit configuration files using the EDTF CL command.

If you set an incorrect value on a configuration file attribute, the value is ignored and an operator message is issued to indicate the problem. (The effect is the same as missing out the attribute entirely.)

Example mqs.ini and qm.ini files

Figure 14 on page 139 shows an example of an mqs.ini file.

```
#*******************************************************************#
#* Module Name: mqs.ini                                            *#
#* Type       : WebSphere MQ Configuration File                    *#
#* Function   : Define WebSphere MQ resources for the node         *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This is an example WebSphere MQ configuration file           *#
#*                                                                 *#
#*******************************************************************#
AllQueueManagers:
#*******************************************************************#
#* The path to the qmgrs directory, below which queue manager data *#
#* is stored                                                       *#
#*******************************************************************#
DefaultPrefix=/QIBM/UserData/mqm

QueueManager:
   Name=saturn.queue.manager
   Prefix=/QIBM/UserData/mqm
   Library=QMSATURN.Q
   Directory=saturn!queue!manager

QueueManager:
   Name=pluto.queue.manager
   Prefix=/QIBM/UserData/mqm
   Library=QMPLUTO.QU
   Directory=pluto!queue!manager

DefaultQueueManager:
   Name=saturn.queue.manager
```

Figure 14. Example of a WebSphere MQ configuration file

Figure 15 on page 140 shows how groups of attributes might be arranged in a queue manager configuration file.

```
#*******************************************************************#
#* Module Name: qm.ini                                             *#
#* Type       : WebSphere MQ queue manager configuration file      *#
#  Function   : Define the configuration of a single queue manager *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This file defines the configuration of the queue manager     *#
#*                                                                 *#
#*******************************************************************#
Log:
   LogPath=QMSATURN.Q
   LogReceiverSize=65536

CHANNELS:
  MaxChannels = 20       ; Maximum number of channels allowed.
                         ; Default is 100.
  MaxActiveChannels = 10 ; Maximum number of channels allowed to be
                         ; active at any time.  The default is the
                         ; value of MaxChannels.

TCP:                     ; TCP/IP entries.
  KeepAlive = Yes        ; Switch KeepAlive on.
  SvrSndBuffSize=20000  ; Size in bytes of the TCP/IP send buffer for each
                         ; channel instance. Default is 32768.
  SvrRcvBuffSize=20000  ; Size in bytes of the TCP/IP receive buffer for each
                         ; channel instance. Default is 32768.
  Connect_Timeout=10000  ; Number of seconds before an attempt to connect the
                         ; channel instance times out. Default is zero (no timeout).

QMErrorLog:
  ErrorLogSize = 262144
  ExcludeMessage = 7234
  SuppressMessage = 9001,9002,9202
  SuppressInterval = 30
```

Figure 15. Example queue manager configuration file

Note:

1. WebSphere MQ on the node uses the default locations for queue managers and the journals.
2. The queue manager saturn.queue.manager is the default queue manager for the node. The directory for files associated with this queue manager has been automatically transformed into a valid file name for the file system.
3. Because the WebSphere MQ configuration file is used to locate the data associated with queue managers, a nonexistent or incorrect configuration file can cause some or all WebSphere MQ commands to fail. Also, applications cannot connect to a queue manager that is not defined in the WebSphere MQ configuration file.

Chapter 10. Installable services and components

This chapter introduces the installable services and the functions and components associated with them. We document the interface to these functions so that you, or software vendors, can supply components.

The chapter includes:

- "Why installable services?"
- "Functions and components" on page 142
- "Initialization" on page 143
- "Configuring services and components" on page 144
- "Creating your own service component" on page 145

The installable services interface is described in "Installable services interface reference information" on page 148.

Why installable services?

The main reasons for providing WebSphere MQ installable services are:

- To provide you with the flexibility of choosing whether to use components provided by WebSphere MQ for i5/OS, or replace or augment them with others.
- To allow vendors to participate, by providing components that might use new technologies, without making internal changes to WebSphere MQ for i5/OS.
- To allow WebSphere MQ to exploit new technologies faster and cheaper, and so provide products earlier and at lower prices.

Installable services and *service components* are part of the WebSphere MQ product structure. At the center of this structure is the part of the queue manager that implements the function and rules associated with the Message Queue Interface (MQI). This central part requires a number of service functions, called *installable services*, in order to perform its work. The installable service available in WebSphere MQ for i5/OS is the authorization service.

Each installable service is a related set of functions implemented using one or more *service components*. Each component is invoked using a properly-architected, publicly-available interface. This enables independent software vendors and other third parties to provide installable components to augment or replace those provided by WebSphere MQ for i5/OS. Table 13 summarizes support for the authorization service.

Table 13. Authorization service components summary

Supplied component	Function	Requirements
Object Authority Manager (OAM)	Provides authorization checking on commands and MQI calls. Users can write their own component to augment or replace the OAM.	(Appropriate platform authorization facilities are assumed)

Table 13. Authorization service components summary (continued)

Supplied component	Function	Requirements
DCE name service component **Note:** DCE is only supported on versions of WebSphere MQ earlier than V6.0.	• Allows queue managers to share queues, or • User defined **Note:** Shared queues must have their *Scope* attribute set to CELL.	• DCE is required for the supplied component, or • A third-party or user-written name manager

Functions and components

Each service consists of a set of related functions. For example, the name service contains function for:

- Looking up a queue name and returning the name of the queue manager where the queue is defined
- Inserting a queue name into the service's directory
- Deleting a queue name from the service's directory

It also contains initialization and termination functions.

An installable service is provided by one or more service components. Each component can perform some or all of the functions that are defined for that service. The component is also responsible for managing any underlying resources or software that it needs to implement the service. Configuration files provide a standard way of loading the component and determining the addresses of the functional routines that it provides.

Services and components are related as follows:

- A service is defined to a queue manager by stanzas in a configuration file.
- Each service is supported by supplied code in the queue manager. Users cannot change this code and therefore cannot create their own services.
- Each service is implemented by one or more components; these can be supplied with the product or user-written. Multiple components for a service can be invoked, each supporting different facilities within the service.
- Entry points connect the service components to the supporting code in the queue manager.

Entry-points

Each service component is represented by a list of the entry-point addresses of the routines that support a particular installable service. The installable service defines the function to be performed by each routine.

The ordering of the service components when they are configured defines the order in which entry-points are called in an attempt to satisfy a request for the service.

In the supplied header file `cmqzc.h`, the supplied entry points to each service have an MQZID_ prefix.

Return codes

Service components provide return codes to the queue manager to report on a variety of conditions. They report the success or failure of the operation, and indicate whether the queue manager is to proceed to the next service component. A separate *Continuation* parameter carries this indication.

Component data

A single service component might require data to be shared between its various functions. Installable services provide an optional data area to be passed on each invocation of a given service component. This data area is for the exclusive use of the service component. It is shared by all the invocations of a given function, even if they are made from different address spaces or processes. It is guaranteed to be addressable from the service component whenever it is called. You must declare the size of this area in the *ServiceComponent* stanza.

Initialization

When the component initialization routine is invoked, it must call the queue manager **MQZEP** function for each entry-point supported by the component. **MQZEP** defines an entry-point to the service. All the undefined exit points are assumed to be NULL.

Primary initialization

A component is always invoked with this option once, before it is invoked in any other way.

Secondary initialization

A component can be invoked with this option on certain platforms. For example, it can be invoked once for each operating system process, thread, or task by which the service is accessed.

If secondary initialization is used:

- The component can be invoked more than once for secondary initialization. For each such call, a matching call for secondary termination is issued when the service is no longer needed.

  For authorization services this is the MQZ_TERM_AUTHORITY call.
- The entry points must be re-specified (by calling MQZEP) each time the component is called for primary and secondary initialization.
- Only one copy of component data is used for the component; there is not a different copy for each secondary initialization.
- The component is not invoked for any other calls to the service (from the operating system process, thread, or task, as appropriate) before secondary initialization has been carried out.
- The component must set the *Version* parameter to the same value for primary and secondary initialization.

Primary termination

The primary termination component is always invoked with this option once, when it is no longer required. No further calls are made to this component.

Secondary termination

The secondary termination component is invoked with this option, if it has been invoked for secondary initialization.

Configuring services and components

Configure service components using the queue manager configuration files. Each service used must have a *Service* stanza, which defines the service to the queue manager.

For each component within a service, there must be a *ServiceComponent* stanza. This identifies the name and path of the module containing the code for that component.

The authorization service component, known as the Object Authority Manager (OAM), is supplied with the product. When you create a queue manager, the queue manager configuration file is automatically updated to include the appropriate stanzas for the authorization service and for the default component (the OAM).

The code for each service component is loaded into the queue manager when the queue manager is started, using dynamic binding, where this is supported on the platform.

Service stanza format

The format of the *Service* stanza is:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
```

where:

<service_name>
The name of the service. This is defined by the service.

<entries>
The number of entry-points defined for the service. This includes the initialization and termination entry points.

Service component stanza format

The format of the *Service component* stanza is:

```
ServiceComponent:
  Service=<service_name>
  Name=<component_name>
  Module=<module_name>
  ComponentDataSize=<size>
```

where:

<service_name>
The name of the service. This must match the `Name` specified in a service stanza.

<component_name>
A descriptive name of the service component. This must be unique, and contain only the characters that are valid for the names of WebSphere MQ objects (for example, queue names). This name occurs in operator messages generated by the service. We recommend that you use a name starting with a company trademark or similar distinguishing string.

<module_name>
The name of the module to contain the code for this component. Specify a full path name.

<size> The size in bytes of the component data area passed to the component on each call. Specify zero if no component data is required.

These two stanzas can appear in any order and the stanza keys under them can also appear in any order. For either of these stanzas, all the stanza keys must be present. If a stanza key is duplicated, the last one is used.

At startup time, the queue manager processes each service component entry in the configuration file in turn. It then loads the specified component module, invoking the entry-point of the component (which must be the entry-point for initialization of the component), passing it a configuration handle.

Creating your own service component

To create your own service component:

- Ensure that the header file `cmqzc.h` is included in your program.
- Create the shared library by compiling the program and linking it with the shared libraries `libmqm*` and `libmqmzf*`.

  Note: Because the agent can run in a threaded environment, you must build the OAM to run in a threaded environment. This includes using the threaded versions of `libmqm` and `libmqmzf`.
- Add stanzas to the queue manager configuration file to define the service to the queue manager and to specify the location of the module. Refer to the individual chapters for each service, for more information.
- Stop and restart the queue manager to activate the component.

Authorization service

The authorization service is an installable service that enables queue managers to invoke authorization facilities, for example, checking that a user ID has authority to open a queue.

This service is a component of the WebSphere MQ security enabling interface (SEI), which is part of the WebSphere MQ framework.

This chapter discusses:

- "Object authority manager (OAM)" on page 146
- "Configuring authorization service stanzas" on page 146
- "Authorization service interface" on page 147

Object authority manager (OAM)

The authorization service component supplied with the WebSphere MQ products is called the Object Authority Manager (OAM). By default, the OAM is active and works with the control commands **WRKMQMAUT** (work with authority), **WRKMQMAUTD** (work with authority data), **DSPMQMAUT** (display object authority), **GRTMQMAUT** (grant object authority), **RVKMQMAUT** (revoke object authority), and **RFRMQMAUT** (refresh security).

The syntax of these commands and how to use them are described in the CL command help.

The OAM works with the *entity* of a principal or group.

When an MQI request is made or a command is issued, the OAM checks the authorization of the entity associated with the operation to see whether it can:
- Perform the requested operation.
- Access the specified queue manager resources.

The authorization service enables you to augment or replace the authority checking provided for queue managers by writing your own authorization service component.

Defining the service to the operating system

The authorization service stanzas in the queue manager configuration file `qm.ini` define the authorization service to the queue manager. See "Configuring services and components" on page 144 for information about the types of stanza.

Configuring authorization service stanzas

On WebSphere MQ for i5/OS:

Principal
Is an i5/OS system user profile.

Group Is an i5/OS system group profile.

Authorizations can be granted or revoked at the group level only. A request to grant or revoke a user's authority updates the primary group for that user.

Each queue manager has its own queue manager configuration file. For example, the default path and file name of the queue manager configuration file for queue manager QMNAME is `/QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini`.

The *`Service`* stanza and the *`ServiceComponent`* stanza for the default authorization component are added to `qm.ini` automatically, but can be overridden by `WRKENVVAR`. Any other *`ServiceComponent`* stanzas must be added manually.

For example, the following stanzas in the queue manager configuration file define two authorization service components:

```
Service:
    Name=AuthorizationService
    EntryPoints=7

ServiceComponent:
    Service=AuthorizationService
    Name=MQ.UNIX.authorization.service
    Module=QMQM/AMQZFU
    ComponentDataSize=0

ServiceComponent:
    Service=AuthorizationService
    Name=user.defined.authorization.service
    Module=LIBRARY/SERVICE PROGRAM NAME
    ComponentDataSize=96
```

Figure 16. WebSphere MQ for i5/OS authorization service stanzas in qm.ini

The first service component stanza (`MQ.UNIX.authorization.service`) defines the default authorization service component, the OAM. If you remove this stanza and restart the queue manager, the OAM is disabled and no authorization checks are made.

Authorization service interface

The authorization service provides the following entry points for use by the queue manager:

MQZ_AUTHENTICATE_USER
Authenticates a user ID and password, and can set identity context fields.

MQZ_CHECK_AUTHORITY
Checks whether an entity has authority to perform one or more operations on a specified object.

MQZ_COPY_ALL_AUTHORITY
Copies all the current authorizations that exist for a referenced object to another object.

MQZ_DELETE_AUTHORITY
Deletes all authorizations associated with a specified object.

MQZ_ENUMERATE_AUTHORITY_DATA
Retrieves all the authority data that matches the selection criteria specified.

MQZ_FREE_USER
Frees associated allocated resources.

MQZ_GET_AUTHORITY
Gets the authority that an entity has to access a specified object.

MQZ_GET_EXPLICIT_AUTHORITY
Gets either the authority that a named group has to access a specified object (but without the additional authority of the **nobody** group) or the authority that the primary group of the named principal has to access a specified object.

MQZ_INIT_AUTHORITY
Initializes authorization service component.

MQZ_INQUIRE
Queries the supported functionality of the authorization service.

MQZ_REFRESH_CACHE
Refresh all authorizations.

MQZ_SET_AUTHORITY
Sets the authority that an entity has to a specified object.

MQZ_TERM_AUTHORITY
Terminates authorization service component.

These entry points support the use of the Windows Security Identifier (NT SID).

These names are defined as **typedef**s, in the header file `cmqzc.h`, which can be used to prototype the component functions.

The initialization function (**MQZ_INIT_AUTHORITY**) must be the main entry point for the component. The other functions are invoked through the entry point address that the initialization function has added into the component entry point vector.

See "Creating your own service component" on page 145 for more information.

Installable services interface reference information

This section provides reference information for the installable services. It includes:

- "How the functions are shown"
- "MQZEP – Add component entry point" on page 149
- "MQZ_AUTHENTICATE_USER – Authenticate user" on page 150
- "MQZ_CHECK_AUTHORITY – Check authority" on page 153
- "MQZ_COPY_ALL_AUTHORITY – Copy all authority" on page 157
- "MQZ_DELETE_AUTHORITY – Delete authority" on page 160
- "MQZ_ENUMERATE_AUTHORITY_DATA – Enumerate authority data" on page 162
- "MQZ_FREE_USER – Free user" on page 165
- "MQZ_GET_AUTHORITY – Get authority" on page 167
- "MQZ_GET_EXPLICIT_AUTHORITY – Get explicit authority" on page 170
- "MQZ_INIT_AUTHORITY – Initialize authorization service" on page 173
- "MQZ_INQUIRE – Inquire authorization service" on page 175
- "MQZ_REFRESH_CACHE – Refresh all authorizations" on page 179
- "MQZ_SET_AUTHORITY – Set authority" on page 181
- "MQZ_TERM_AUTHORITY – Terminate authorization service" on page 184
- "MQZAC – Application context" on page 185
- "MQZAD – Authority data" on page 188
- "MQZED – Entity descriptor" on page 191
- "MQZFP – Free parameters" on page 192
- "MQZIC – Identity context" on page 193

The functions and data types are in alphabetic order within the group for each service type.

How the functions are shown

For each function there is a description, including the function identifier (for MQZEP).

The *parameters* are shown listed in the order they must occur. They must all be present.

Parameters and data types

Each parameter name is followed by its data type in parentheses. These are the elementary data types described in the WebSphere MQ Application Programming Reference manual.

The C language invocation is also given, after the description of the parameters.

MQZEP – Add component entry point

This function is invoked by a service component, during initialization, to add an entry point to the entry point vector for that service component.

Syntax

MQZEP *(Hconfig, Function, EntryPoint, CompCode, Reason)*

Parameters

The MQZEP call has the following parameters.

Hconfig (MQHCONFIG) – input:

Configuration handle.

This handle represents the component which is being configured for this particular installable service. It must be the same as the one passed to the component configuration function by the queue manager on the component initialization call.

Function (MQLONG) – input:

Function identifier.

Valid values for this are defined for each installable service.

If MQZEP is called more than once for the same function, the last call made provides the entry point which is used.

EntryPoint (PMQFUNC) – input:

Function entry point.

This is the address of the entry point provided by the component to perform the function.

The value NULL is valid, and indicates that the function is not provided by this component. NULL is assumed for entry points which are not defined using MQZEP.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:

MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_FUNCTION_ERROR
(2281, X'8E9') Function identifier not valid.
MQRC_HCONFIG_ERROR
(2280, X'8E8') Configuration handle not valid.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

C invocation

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHCONFIG  Hconfig;     /* Configuration handle */
MQLONG     Function;    /* Function identifier */
PMQFUNC    EntryPoint;  /* Function entry point */
MQLONG     CompCode;    /* Completion code */
MQLONG     Reason;      /* Reason code qualifying CompCode */
```

MQHCONFIG – Configuration handle

The MQHCONFIG data type represents a configuration handle, that is, the component that is being configured for a particular installable service. A configuration handle must be aligned on its natural boundary.

Note: Applications must test variables of this type for equality only.

C declaration

```
typedef void MQPOINTER MQHCONFIG;
```

PMQFUNC – Pointer to function

Pointer to a function.

C declaration

```
typedef void MQPOINTER PMQFUNC;
```

MQZ_AUTHENTICATE_USER – Authenticate user

This function is provided by a MQZAS_VERSION_5 authorization service component, and is invoked by the queue manager to authenticate a user, or to set identity context fields. It is invoked when WebSphere MQ's user application context is established. This happens during connect calls at the point where the

application's user context is initialized, and at each point where the application's user context is changed. Each time a connect call is made, the application's user context information is reacquired in the *IdentityContext* field.

The function identifier for this function (for MQZEP) is MQZID_AUTHENTICATE_USER.

Syntax

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
        IdentityContext, CorrelationPtr, ComponentData, Continuation, CompCode, Reason)
```

Parameters

The MQZ_AUTHENTICATE_USER call has the following parameters.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

SecurityParms (MQCSP) – input:

Security parameters.

Data relating to the user ID, password, and authentication type. If the *AuthenticationType* attribute of the MQCSP structure is specified as MQCSP_AUTH_USER_ID_AND_PWD, both the user ID and password are compared against the equivalent fields in the *IdentityContext* (MQZIC) parameter to determine whether they match. For more information, see the WebSphere MQ Application Programming Reference.

During an MQCONN MQI call this parameter contains null, or default values.

ApplicationContext (MQZAC) – input:

Application context.

Data relating to the calling application. See "MQZAC – Application context" on page 185 for details.

During every MQCONN or MQCONNX MQI call, the user context information in the MQZAC structure is reacquired.

IdentityContext (MQZIC) – input/output:

Identity context.

On input to the authenticate user function, this identifies the current identity context. The authenticate user function can change this, at which point the queue manager adopts the new identity context. See "MQZIC – Identity context" on page 193 for more details on the MQZIC structure.

CorrelationPtr (MQPTR) – output:

Correlation pointer.

Specifies the address of any correlation data. This pointer is subsequently passed on to other OAM calls.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Continuation (MQLONG) – output:

Continuation flag.

The following values can be specified:

MQZCI_DEFAULT
Continuation dependent on other components.

MQZCI_STOP
Do not continue with next component.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:

MQCC_OK
Successful completion.

MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.

C invocation

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                  IdentityContext, &CorrelationPtr, ComponentData,
                  &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCSP     SecurityParms;      /* Security parameters */
MQZAC     ApplicationContext; /* Application context */
MQZIC     IdentityContext;    /* Identity context */
MQPTR     CorrelationPtr;     /* Correlation pointer */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                                 component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_CHECK_AUTHORITY – Check authority

This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to check whether an entity has authority to perform a particular action, or actions, on a specified object.

The function identifier for this function (for MQZEP) is MQZID_CHECK_AUTHORITY.

Syntax

MQZ_CHECK_AUTHORITY *(QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

Parameters

The MQZ_CHECK_AUTHORITY call has the following parameters.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

EntityName (MQCHAR12) – input:

Entity name.

The name of the entity whose authorization to the object is to be checked. The maximum length of the string is 12 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

It is not essential for this entity to be known to the underlying security service. If it is not known, the authorizations of the special **nobody** group (to which all entities are assumed to belong) are used for the check. An all-blank name is valid and can be used in this way.

EntityType (MQLONG) – input:

Entity type.

The type of entity specified by *EntityName*. It is one of the following:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Group.

ObjectName (MQCHAR48) – input:

Object name.

The name of the object to which access is required. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

ObjectType (MQLONG) – input:

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO
Authentication information.

MQOT_CHANNEL
Channel.

MQOT_CLNTCONN_CHANNEL
Client connection channel.

MQOT_LISTENER
Listener.

MQOT_NAMELIST
Namelist.

MQOT_PROCESS
Process definition.

MQOT_Q
Queue.

MQOT_Q_MGR
Queue manager.

MQOT_SERVICE
Service.

Authority (MQLONG) – input:

Authority to be checked.

If one authorization is being checked, this field is equal to the appropriate authorization operation (MQZAO_* constant). If more than one authorization is being checked, it is the bitwise OR of the corresponding MQZAO_* constants.

The following authorizations apply to use of the MQI calls:

MQZAO_CONNECT
Ability to use the MQCONN call.

MQZAO_BROWSE
Ability to use the MQGET call with a browse option.

This allows the MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR, or MQGMO_BROWSE_NEXT option to be specified on the MQGET call.

MQZAO_INPUT
Ability to use the MQGET call with an input option.

This allows the MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE, or MQOO_INPUT_AS_Q_DEF option to be specified on the MQOPEN call.

MQZAO_OUTPUT
Ability to use the MQPUT call.

This allows the MQOO_OUTPUT option to be specified on the MQOPEN call.

MQZAO_INQUIRE
Ability to use the MQINQ call.

This allows the MQOO_INQUIRE option to be specified on the MQOPEN call.

MQZAO_SET
Ability to use the MQSET call.

This allows the MQOO_SET option to be specified on the MQOPEN call.

MQZAO_PASS_IDENTITY_CONTEXT
Ability to pass identity context.

This allows the MQOO_PASS_IDENTITY_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_PASS_IDENTITY_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_PASS_ALL_CONTEXT
Ability to pass all context.

This allows the MQOO_PASS_ALL_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_PASS_ALL_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_SET_IDENTITY_CONTEXT
Ability to set identity context.

This allows the MQOO_SET_IDENTITY_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_SET_IDENTITY_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_SET_ALL_CONTEXT
Ability to set all context.

This allows the MQOO_SET_ALL_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_SET_ALL_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_ALTERNATE_USER_AUTHORITY
Ability to use alternate user authority.

This allows the MQOO_ALTERNATE_USER_AUTHORITY option to be specified on the MQOPEN call, and the MQPMO_ALTERNATE_USER_AUTHORITY option to be specified on the MQPUT1 call.

MQZAO_ALL_MQI
All of the MQI authorizations.

This enables all of the authorizations described above.

The following authorizations apply to administration of a queue manager:

MQZAO_CREATE
Ability to create objects of a specified type.

MQZAO_DELETE
Ability to delete a specified object.

MQZAO_DISPLAY
Ability to display the attributes of a specified object.

MQZAO_CHANGE
Ability to change the attributes of a specified object.

MQZAO_CLEAR
Ability to delete all messages from a specified queue.

MQZAO_AUTHORIZE
Ability to authorize other users for a specified object.

MQZAO_CONTROL
Ability to start, stop, or ping a non-client channel object.

MQZAO_CONTROL_EXTENDED
Ability to reset a sequence number, or resolve an indoubt message on a non-client channel object.

MQZAO_ALL_ADMIN
All of the administration authorizations, other than MQZAO_CREATE.

The following authorizations apply to both use of the MQI and to administration of a queue manager:

MQZAO_ALL
All authorizations, other than MQZAO_CREATE.

MQZAO_NONE
No authorizations.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Continuation (MQLONG) – output:

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT
Continuation dependent on queue manager.

For MQZ_CHECK_AUTHORITY this has the same effect as MQZCI_STOP.

MQZCI_CONTINUE
Continue with next component.
MQZCI_STOP
Do not continue with next component.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_NOT_AUTHORIZED
(2035, X'7F3') Not authorized for access.
MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.
MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Underlying service not available.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

C invocation

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_COPY_ALL_AUTHORITY – Copy all authority

This function is provided by an authorization service component. It is invoked by the queue manager to copy all of the authorizations that are currently in force for a reference object to another object.

The function identifier for this function (for MQZEP) is MQZID_COPY_ALL_AUTHORITY.

Syntax

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName,
       ObjectType, ComponentData, Continuation, CompCode, Reason)
```

Parameters

The MQZ_COPY_ALL_AUTHORITY call has the following parameters.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

RefObjectName (MQCHAR48) – input:

Reference object name.

The name of the reference object, the authorizations for which are to be copied. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

ObjectName (MQCHAR48) – input:

Object name.

The name of the object for which accesses are to be set. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

ObjectType (MQLONG) – input:

Object type.

The type of object specified by *RefObjectName* and *ObjectName*. It is one of the following:

MQOT_AUTH_INFO
Authentication information.

MQOT_CHANNEL
Channel.

MQOT_CLNTCONN_CHANNEL
Client connection channel.

MQOT_LISTENER
Listener.

MQOT_NAMELIST
Namelist.

MQOT_PROCESS
Process definition.

MQOT_Q
Queue.

MQOT_Q_MGR
Queue manager.

MQOT_SERVICE
Service.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Continuation (MQLONG) – output:

Continuation indicator set by component.

The following values can be specified:
MQZCI_DEFAULT
Continuation dependent on queue manager.

For MQZ_COPY_ALL_AUTHORITY this has the same effect as MQZCI_STOP.
MQZCI_CONTINUE
Continue with next component.
MQZCI_STOP
Do not continue with next component.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Underlying service not available.

MQRC_UNKNOWN_REF_OBJECT
(2294, X'8F6') Reference object unknown.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

C invocation

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  RefObjectName;     /* Reference object name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_DELETE_AUTHORITY – Delete authority

This function is provided by an authorization service component, and is invoked by the queue manager to delete all of the authorizations associated with the specified object.

The function identifier for this function (for MQZEP) is MQZID_DELETE_AUTHORITY.

Syntax

MQZ_DELETE_AUTHORITY *(QMgrName, ObjectName, ObjectType, ComponentData, Continuation, CompCode, Reason)*

Parameters

The MQZ_DELETE_AUTHORITY call has the following parameters.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

ObjectName (MQCHAR48) – input:

Object name.

The name of the object for which accesses are to be deleted. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

ObjectType (MQLONG) – input:

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO
Authentication information.

MQOT_CHANNEL
Channel.

MQOT_CLNTCONN_CHANNEL
Client connection channel.

MQOT_LISTENER
Listener.

MQOT_NAMELIST
Namelist.

MQOT_PROCESS
Process definition.

MQOT_Q
Queue.

MQOT_Q_MGR
Queue manager.

MQOT_SERVICE
Service.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Continuation (MQLONG) – output:

Continuation indicator set by component.

The following values can be specified:
MQZCI_DEFAULT
Continuation dependent on queue manager.

For MQZ_DELETE_AUTHORITY this has the same effect as MQZCI_STOP.

MQZCI_CONTINUE
Continue with next component.

MQZCI_STOP
Do not continue with next component.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:

MQCC_OK
Successful completion.

MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Underlying service not available.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

C invocation

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_ENUMERATE_AUTHORITY_DATA – Enumerate authority data

This function is provided by an MQZAS_VERSION_4 authorization service component, and is invoked repeatedly by the queue manager to retrieve all of the authority data that matches the selection criteria specified on the first invocation.

The function identifier for this function (for MQZEP) is MQZID_ENUMERATE_AUTHORITY_DATA.

Syntax

MQZ_ENUMERATE_AUTHORITY_DATA *(QMgrName, StartEnumeration, Filter, AuthorityBufferLength, AuthorityBuffer, AuthorityDataLength, ComponentData, Continuation, CompCode, Reason)*

Parameters

The MQZ_ENUMERATE_AUTHORITY_DATA call has the following parameters.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

StartEnumeration (MQLONG) – input:

Flag indicating whether call should start enumeration.

This indicates whether the call should start the enumeration of authority data, or continue the enumeration of authority data started by a previous call to MQZ_ENUMERATE_AUTHORITY_DATA. The value is one of the following:

MQZSE_START
Start enumeration.

The call is invoked with this value to start the enumeration of authority data. The *Filter* parameter specifies the selection criteria to be used to select the authority data returned by this and successive calls.

MQZSE_CONTINUE
Continue enumeration.

The call is invoked with this value to continue the enumeration of authority data. The *Filter* parameter is ignored in this case, and can be specified as the null pointer (the selection criteria are determined by the *Filter* parameter specified by the call that had *StartEnumeration* set to MQZSE_START).

Filter (MQZAD) – input:

Filter.

If *StartEnumeration* is MQZSE_START, *Filter* specifies the selection criteria to be used to select the authority data to return. If *Filter* is the null pointer, no selection criteria are used, that is, all authority data is returned. See "MQZAD – Authority data" on page 188 for details of the selection criteria that can be used.

If *StartEnumeration* is MQZSE_CONTINUE, *Filter* is ignored, and can be specified as the null pointer.

AuthorityBufferLength (MQLONG) – input:

Length of *AuthorityBuffer*.

This is the length in bytes of the *AuthorityBuffer* parameter. The authority buffer must be big enough to accommodate the data to be returned.

AuthorityBuffer (MQZAD) – output:

Authority data.

This is the buffer in which the authority data is returned. The buffer must be big enough to accommodate an MQZAD structure, an MQZED structure, plus the longest entity name and longest domain name defined.

Note: This parameter is defined as an MQZAD, as the MQZAD always occurs at the start of the buffer. However, if the buffer is actually declared as an MQZAD, the buffer will be too small – it needs to be bigger than an MQZAD so that it can accommodate the MQZAD, MQZED, plus entity and domain names.

AuthorityDataLength (MQLONG) – output:

Length of data returned in *AuthorityBuffer*.

This is the length of the data returned in *AuthorityBuffer*. If the authority buffer is too small, *AuthorityDataLength* is set to the length of the buffer required, and the call returns completion code MQCC_FAILED and reason code MQRC_BUFFER_LENGTH_ERROR.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Continuation (MQLONG) – output:

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT
: Continuation dependent on queue manager.

  For MQZ_ENUMERATE_AUTHORITY_DATA this has the same effect as MQZCI_CONTINUE.

MQZCI_CONTINUE
: Continue with next component.

MQZCI_STOP
: Do not continue with next component.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Buffer length parameter not valid.
MQRC_NO_DATA_AVAILABLE
(2379, X'94B') No data available.
MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

C invocation

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                              AuthorityBufferLength,
                              &AuthorityBuffer,
                              &AuthorityDataLength, ComponentData,
                              &Continuation, &CompCode,
                              &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;               /* Queue manager name */
MQLONG    StartEnumeration;       /* Flag indicating whether call should
                                     start enumeration */
MQZAD     Filter;                 /* Filter */
MQLONG    AuthorityBufferLength;  /* Length of AuthorityBuffer */
MQZAD     AuthorityBuffer;        /* Authority data */
MQLONG    AuthorityDataLength;    /* Length of data returned in
                                     AuthorityBuffer */
MQBYTE    ComponentData[n];       /* Component data */
MQLONG    Continuation;           /* Continuation indicator set by
                                     component */
MQLONG    CompCode;               /* Completion code */
MQLONG    Reason;                 /* Reason code qualifying CompCode */
```

MQZ_FREE_USER – Free user

This function is provided by a MQZAS_VERSION_5 authorization service component, and is invoked by the queue manager to free associated allocated resource. It is invoked when an application has finished running under all user contexts, for example during an MQDISC MQI call.

The function identifier for this function (for MQZEP) is MQZID_FREE_USER.

Syntax

```
MQZ_FREE_USER (QMgrName, FreeParms, ComponentData, Continuation,
       CompCode, Reason)
```

Parameters

The MQZ_FREE_USER call has the following parameters.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

FreeParms (MQZFP) – input:

Free parameters.

A structure containing data relating to the resource to be freed. See "MQZFP – Free parameters" on page 192 for details.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

Continuation (MQLONG) – output:

Continuation flag.

The following values can be specified:

MQZCI_DEFAULT
Continuation dependent on other components.

MQZCI_STOP
Do not continue with next component.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:

MQCC_OK
Successful completion.

MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.

C invocation

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                       IdentityContext, CorrelationPtr, ComponentData,
                       &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZFP     FreeParms;          /* Resource to be freed */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                                 component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

MQZ_GET_AUTHORITY – Get authority

This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to retrieve the authority that an entity has to access the specified object.

The function identifier for this function (for MQZEP) is MQZID_GET_AUTHORITY.

Syntax

MQZ_GET_AUTHORITY *(QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

Parameters

The MQZ_GET_AUTHORITY call has the following parameters.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

EntityName (MQCHAR12) – input:

Entity name.

The name of the entity whose access to the object is to be retrieved. The maximum length of the string is 12 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

EntityType (MQLONG) – input:

Entity type.

The type of entity specified by *EntityName*. The following value can be specified:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Group.

ObjectName (MQCHAR48) – input:

Object name.

The name of the object for which the entity's authority is to be retrieved. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

ObjectType (MQLONG) – input:

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO
Authentication information.

MQOT_CHANNEL
Channel.

MQOT_CLNTCONN_CHANNEL
Client connection channel.

MQOT_LISTENER
Listener.

MQOT_NAMELIST
Namelist.

MQOT_PROCESS
Process definition.

MQOT_Q
Queue.

MQOT_Q_MGR
Queue manager.

MQOT_SERVICE
Service.

Authority (MQLONG) – output:

Authority of entity.

If the entity has one authority, this field is equal to the appropriate authorization operation (MQZAO_* constant). If it has more than one authority, this field is the bitwise OR of the corresponding MQZAO_* constants.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Continuation (MQLONG) – output:

Continuation indicator set by component.

The following values can be specified:
MQZCI_DEFAULT
Continuation dependent on queue manager.

For MQZ_GET_AUTHORITY this has the same effect as MQZCI_CONTINUE.
MQZCI_CONTINUE
Continue with next component.
MQZCI_STOP
Do not continue with next component.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_NOT_AUTHORIZED
(2035, X'7F3') Not authorized for access.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Underlying service not available.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entity unknown to service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

C invocation

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, &Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_GET_EXPLICIT_AUTHORITY – Get explicit authority

This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to retrieve the authority that a named group has to access a specified object (but without the additional authority of the **nobody** group), or the authority that the primary group of the named principal has to access a specified object.

The function identifier for this function (for MQZEP) is MQZID_GET_EXPLICIT_AUTHORITY.

Syntax

MQZ_GET_EXPLICIT_AUTHORITY *(QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

Parameters

The MQZ_GET_EXPLICIT_AUTHORITY call has the following parameters.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

EntityName (MQCHAR12) – input:

Entity name.

The name of the entity whose access to the object is to be retrieved. The maximum length of the string is 12 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

EntityType (MQLONG) – input:

Entity type.

The type of entity specified by *EntityName*. The following value can be specified:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Group.

ObjectName (MQCHAR48) – input:

Object name.

The name of the object for which the entity's authority is to be retrieved. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

ObjectType (MQLONG) – input:

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO
Authentication information.

MQOT_CHANNEL
Channel.

MQOT_CLNTCONN_CHANNEL
Client connection channel.

MQOT_LISTENER
Listener.

MQOT_NAMELIST
Namelist.

MQOT_PROCESS
Process definition.

MQOT_Q
Queue.

MQOT_Q_MGR
Queue manager.

MQOT_SERVICE
Service.

Authority (MQLONG) – output:

Authority of entity.

If the entity has one authority, this field is equal to the appropriate authorization operation (MQZAO_* constant). If it has more than one authority, this field is the bitwise OR of the corresponding MQZAO_* constants.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Continuation (MQLONG) – output:

Continuation indicator set by component.

The following values can be specified:
MQZCI_DEFAULT
Continuation dependent on queue manager.

For MQZ_GET_EXPLICIT_AUTHORITY this has the same effect as MQZCI_CONTINUE.
MQZCI_CONTINUE
Continue with next component.
MQZCI_STOP
Do not continue with next component.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_NOT_AUTHORIZED
(2035, X'7F3') Not authorized for access.
MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.
MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Underlying service not available.
MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entity unknown to service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

C invocation

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                            ObjectName, ObjectType, &Authority,
                            ComponentData, &Continuation,
                            &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_INIT_AUTHORITY – Initialize authorization service

This function is provided by an authorization service component, and is invoked by the queue manager during configuration of the component. It is expected to call MQZEP in order to provide information to the queue manager.

The function identifier for this function (for MQZEP) is MQZID_INIT_AUTHORITY.

Syntax

MQZ_INIT_AUTHORITY *(Hconfig, Options, QMgrName, ComponentDataLength, ComponentData, Version, CompCode, Reason)*

Parameters

The MQZ_INIT_AUTHORITY call has the following parameters.

Hconfig (MQHCONFIG) – input:

Configuration handle.

This handle represents the particular component being initialized. It is to be used by the component when calling the queue manager with the MQZEP function.

Options (MQLONG) – input:

Initialization options.

It is one of the following:
MQZIO_PRIMARY
Primary initialization.
MQZIO_SECONDARY
Secondary initialization.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

ComponentDataLength (MQLONG) – input:

Length of component data.

Length in bytes of the *ComponentData* area. This length is defined in the component configuration data.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This is initialized to all zeroes before calling the component's primary initialization function. This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions (including the initialization function) provided by this component are preserved, and presented the next time one of this component's functions is called.

Version (MQLONG) – input/output:

Version number.

On input to the initialization function, this identifies the *highest* version number that the queue manager supports. The initialization function must change this, if necessary, to the version of the interface which *it* supports. If on return the queue manager does not support the version returned by the component, it calls the component's MQZ_TERM_AUTHORITY function and makes no further use of this component.

The following values are supported:
MQZAS_VERSION_1
Version 1.

MQZAS_VERSION_2
Version 2.
MQZAS_VERSION_3
Version 3.
MQZAS_VERSION_4
Version 4.
MQZAS_VERSION_5
Version 5.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_INITIALIZATION_FAILED
(2286, X'8EE') Initialization failed for an undefined reason.
MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Underlying service not available.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

C invocation

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

The parameters passed to the service are declared as follows:

```
MQHCONFIG  Hconfig;              /* Configuration handle */
MQLONG     Options;              /* Initialization options */
MQCHAR48   QMgrName;             /* Queue manager name */
MQLONG     ComponentDataLength;  /* Length of component data */
MQBYTE     ComponentData[n];     /* Component data */
MQLONG     Version;              /* Version number */
MQLONG     CompCode;             /* Completion code */
MQLONG     Reason;               /* Reason code qualifying CompCode */
```

MQZ_INQUIRE – Inquire authorization service

This function is provided by a MQZAS_VERSION_5 authorization service component, and is invoked by the queue manager to query the supported functionality. Where multiple service components are used, service components are called in reverse order to the order they were installed in.

The function identifier for this function (for MQZEP) is MQZID_INQUIRE.

Syntax

```
MQZ_INQUIRE
        (QMgrName, SelectorCount, Selectors, IntAttrCount, IntAttrs, CharAttrLength, CharAttrs,
        SelectorReturned, ComponentData, Continuation, CompCode, Reason)
```

Parameters

The MQZ_INQUIRE call has the following parameters.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

SelectorCount (MQLONG) – input:

Number of selectors.

The number of selectors supplied in the Selectors parameter.

The value must be between zero and 256.

Selectors (MQLONG×SelectorCount) – input:

Selectors.

Array of selectors. Each selector identifies a required attribute and must be of one of the following types:

- MQIACF_* (integer)
- MQCACF_* (character)

Selectors can be specified in any order. The number of selectors in the array is indicated by the SelectorCount parameter.

Integer attributes identified by selectors are returned in the IntAttrs parameter in the same order as they appear in Selectors.

Character attributes identified by selectors are returned in the CharAttrs parameter in the same order as they in appear Selectors.

IntAttrCount (MQLONG) – input:

Number of integer attributes.

The number of integer attributes supplied in the IntAttrs parameter.

The value must be between zero and 256.

IntAttrs (MQLONG×IntAttrCount) – output:

Integer attributes.

Array of integer attributes. The integer attributes are returned in the same order as the corresponding integer selectors in the Selectors array.

CharAttrCount (MQLONG) – input:

Length of the character attributes buffer.

The length in bytes of the CharAttrs parameter.

The value must at least sum of the lengths of the requested character attributes. If no character attributes are requested, zero is a valid value.

CharAttrs (MQLONG×CharAttrCount) – output:

Character attributes buffer.

Buffer containing character attributes, concatenated together. The character attributes are returned in the same order as the corresponding character selectors in the Selectors array.

The length of the buffer is given by the CharAttrCount parameter.

SelectorReturned (MQLONG×SelectorCount) – input:

Selector returned.

Array of values identifying which attributes have been returned from the set requested for by the selectors in the Selectors parameter. The number of values in this array is indicated by the SelectorCount parameter. Each value in the array relates to the selector from the corresponding position in the Selectors array. Each value is one of the following:

MQZSL_RETURNED
The attribute requested by the corresponding selector in the Selectors parameter has been returned.

MQZSL_NOT_RETURNED
The attribute requested by the corresponding selector in the Selectors parameter has not been returned.

The array is initialized with all values as *MQZSL_NOT_RETURNED*. When an authorization service component returns an attribute, it sets the appropriate value in the array to *MQZSL_RETURNED*. This allows any other authorization service components, to which the inquire call is made, to identify which attributes have already been returned.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Continuation (MQLONG) – output:

Continuation flag.

The following values can be specified:

MQZCI_DEFAULT
Continuation dependent on other components.

MQZCI_STOP
Do not continue with next component.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:

MQCC_OK
Successful completion.

MQCC_WARNING
Partial completion.

MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
Not enough space for character attributes.

MQRC_INT_COUNT_TOO_SMALL
Not enough space for integer attributes.

If *CompCode* is MQCC_FAILED:

MQRC_SELECTOR_COUNT_ERROR
Number of selectors is not valid.

MQRC_SELECTOR_ERROR
Attribute selector not valid.

MQRC_SELECTOR_LIMIT_EXCEEDED
Too many selectors specified.

MQRC_INT_ATTR_COUNT_ERROR
Number of integer attributes is not valid.

MQRC_INT_ATTRS_ARRAY_ERROR
Integer attributes array not valid.

MQRC_CHAR_ATTR_LENGTH_ERROR
Number of character attributes is not valid.

MQRC_CHAR_ATTRS_ERROR
Character attributes string is not valid.

MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.

C invocation

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
                  &IntAttrs, CharAttrLength, &CharAttrs,
                  SelectorReturned, ComponentData, &Continuation,
                  &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;              /* Queue manager name */
MQLONG     SelectorCount;         /* Selector count */
MQLONG     Selectors[n];          /* Selectors */
MQLONG     IntAttrCount;          /* IntAttrs count */
MQLONG     IntAttrs[n];           /* Integer attributes */
MQLONG     CharAttrCount;         /* CharAttrs count */
MQLONG     CharAttrs[n];          /* Chatacter attributes */
MQLONG     SelectorReturned[n];   /* Selector returned */
MQBYTE     ComponentData[n];      /* Component data */
MQLONG     Continuation;          /* Continuation indicator set by
                                     component */
MQLONG     CompCode;              /* Completion code */
MQLONG     Reason;                /* Reason code qualifying CompCode */
```

MQZ_REFRESH_CACHE – Refresh all authorizations

This function is provided by an MQZAS_VERSION_3 authorization service component, and is invoked by the queue manager to refresh the list of authorizations held internally by the component.

The function identifier for this function (for MQZEP) is MQZID_REFRESH_CACHE (8L).

Syntax

MQZ_REFRESH_CACHE
(QMgrName, ComponentData, Continuation, CompCode, Reason)

Parameters

QMgrName **(MQCHAR48) — input**
Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

ComponentData **(MQBYTE×*ComponentDataLength*) — input/output**
Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Continuation **(MQLONG) — output**
Continuation indicator set by component.

The following values can be specified:
MQZCI_DEFAULT
Continuation dependent on queue manager.

For MQZ_REFRESH_CACHE this has the same effect as MQZCI_CONTINUE.
MQZCI_CONTINUE
Continue with next component.
MQZCI_STOP
Do not continue with next component.

CompCode **(MQLONG) — output**
Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason **(MQLONG) — output**
Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.

For more information on this reason code, see the WebSphere MQ Application Programming Reference book.

C invocation

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_SET_AUTHORITY – Set authority

This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to set the authority that an entity has to access the specified object.

The function identifier for this function (for MQZEP) is MQZID_SET_AUTHORITY.

Note: This function overrides any existing authorities. To preserve any existing authorities you must set them again with this function.

Syntax

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
       ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)
```

Parameters

The MQZ_SET_AUTHORITY call has the following parameters.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

EntityName (MQCHAR12) – input:

Entity name.

The name of the entity whose access to the object is to be set. The maximum length of the string is 12 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

EntityType (MQLONG) – input:

Entity type.

The type of entity specified by *EntityName*. The following value can be specified:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Group.

ObjectName (MQCHAR48) – input:

Object name.

The name of the object to which access is required. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

ObjectType (MQLONG) – input:

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO
Authentication information.

MQOT_CHANNEL
Channel.

MQOT_CLNTCONN_CHANNEL
Client connection channel.

MQOT_LISTENER
Listener.

MQOT_NAMELIST
Namelist.

MQOT_PROCESS
Process definition.

MQOT_Q
Queue.

MQOT_Q_MGR
Queue manager.

MQOT_SERVICE
Service.

Authority (MQLONG) – input:

Authority to be checked.

If one authorization is being set, this field is equal to the appropriate authorization operation (MQZAO_* constant). If more than one authorization is being set, it is the bitwise OR of the corresponding MQZAO_* constants.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Continuation (MQLONG) – output:

Continuation indicator set by component.

The following values can be specified:
MQZCI_DEFAULT
Continuation dependent on queue manager.

For MQZ_SET_AUTHORITY this has the same effect as MQZCI_STOP.
MQZCI_CONTINUE
Continue with next component.
MQZCI_STOP
Do not continue with next component.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_NOT_AUTHORIZED
(2035, X'7F3') Not authorized for access.
MQRC_SERVICE_ERROR
(2289, X'8F1') Unexpected error occurred accessing service.
MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Underlying service not available.
MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Entity unknown to service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

C invocation

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_TERM_AUTHORITY – Terminate authorization service

This function is provided by an authorization service component, and is invoked by the queue manager when it no longer requires the services of this component. The function must perform any cleanup required by the component.

The function identifier for this function (for MQZEP) is MQZID_TERM_AUTHORITY.

Syntax

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
        CompCode, Reason)
```

Parameters

The MQZ_TERM_AUTHORITY call has the following parameters.

Hconfig (MQHCONFIG) – input:

Configuration handle.

This handle represents the particular component being terminated.

Options (MQLONG) – input:

Termination options.

It is one of the following:
MQZTO_PRIMARY
Primary termination.
MQZTO_SECONDARY
Secondary termination.

QMgrName (MQCHAR48) – input:

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

ComponentData (MQBYTE×ComponentDataLength) – input/output:

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter on the MQZ_INIT_AUTHORITY call.

When the MQZ_TERM_AUTHORITY call has completed, the queue manager discards this data.

CompCode (MQLONG) – output:

Completion code.

It is one of the following:
MQCC_OK
Successful completion.
MQCC_FAILED
Call failed.

Reason (MQLONG) – output:

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:
MQRC_NONE
(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:
MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Underlying service not available.
MQRC_TERMINATION_FAILED
(2287, X'8FF') Termination failed for an undefined reason.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

C invocation

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQHCONFIG  Hconfig;           /* Configuration handle */
MQLONG     Options;           /* Termination options */
MQCHAR48   QMgrName;          /* Queue manager name */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     CompCode;          /* Completion code */
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

MQZAC – Application context

The MQZAC structure is used on the MQZ_AUTHENTICATE_USER call for the *ApplicationContext* parameter. This parameter specifies data related to the calling application.

Fields

StrucId (MQCHAR4):

Structure identifier.

The value is:

MQZAC_STRUC_ID
Identifier for application context structure.

For the C programming language, the constant MQZAC_STRUC_ID_ARRAY is also defined; this has the same value as MQZAC_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the service.

Version (MQLONG):

Structure version number.

The value is:

MQZAC_VERSION_1
Version-1 application context structure.

The following constant specifies the version number of the current version:

MQZAC_CURRENT_VERSION
Current version of application context structure.

This is an input field to the service.

ProcessId (MQPID):

Process identifier.

The process identifier of the application.

ThreadId (MQTID):

Thread identifier.

The thread identifier of the application.

ApplName (MQCHAR28):

Application name.

The application name.

UserID (MQCHAR12):

User identifier.

For UNIX systems the application's real user ID. For Windows the application's user ID. For i5/OS systems the user profile that the application job was created under. (i5/OS: when a profile swap is done with the QWTSETP API in the application job the current user profile is returned).

EffectiveUserID (MQCHAR12):

Effective user identifier.

For UNIX systems the application's effective user ID. For Windows this field is all blank. For i5/OS systems the application job's current user profile.

Environment (MQLONG):

Environment.

This field specifies the environment from which the call was made.

The value is one of the following:

MQXE_COMMAND_SERVER
Command server.

MQXE_MQSC
runmqsc command interpreter.

MQXE_MCA
Message channel agent

MQXE_OTHER
Undefined environment

CallerType (MQLONG):

Caller Type.

This field specifies the type of program that made the call.

The value is one of the following:

MQXACT_EXTERNAL
The call is external to the queue manager.

MQXACT_INTERNAL
The call is internal to the queue manager.

AuthenticationType (MQLONG):

Authentication Type.

This field specifies the type of authentication being performed.

The value is one of the following:

MQZAT_INITIAL_CONTEXT
The authentication call is due to user context being initialized. This value is used during an **MQCONN** or **MQCONNX** call.

MQZAT_CHANGE_CONTEXT
The authentication call is due to the user context being changed. This value is used when the MCA changes the user context.

BindType (MQLONG):

Bind Type.

This field specifies the type of binding in use.

The value is one of the following:

MQCNO_FASTPATH_BINDING
Fastpath binding.

MQCNO_SHARED_BINDING
Shared binding.

MQCNO_ISOLATED_BINDING
Isolated binding.

C declaration

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
  MQCHAR4   StrucId;            /* Structure identifier */
  MQLONG    Version;            /* Structure version number */
  MQPID     ProcessId;          /* Process identifier */
  MQTID     ThreadId;           /* Thread identifier */
  MQCHAR28  ApplName;           /* Application name */
  MQCHAR12  UserID;             /* User identifier */
  MQCHAR12  EffectiveUserID;    /* Effective user identifier */
  MQLONG    Environment;        /* Environment */
  MQLONG    CallerType;         /* Caller type */
  MQLONG    AuthenticationType; /* Authentication type */
  MQLONG    BindType;           /* Bind type */
};
```

MQZAD – Authority data

The MQZAD structure is used on the MQZ_ENUMERATE_AUTHORITY_DATA call for two parameters:

- MQZAD is used for the *Filter* parameter which is input to the call. This parameter specifies the selection criteria that are to be used to select the authority data returned by the call.
- MQZAD is also used for the *AuthorityBuffer* parameter which is output from the call. This parameter specifies the authorizations for one combination of profile name, object type, and entity.

Fields

StrucId (MQCHAR4):

Structure identifier.

The value is:

MQZAD_STRUC_ID
Identifier for authority data structure.

For the C programming language, the constant MQZAD_STRUC_ID_ARRAY is also defined; this has the same value as MQZAD_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the service.

Version (MQLONG):

Structure version number.

The value is:

MQZAD_VERSION_1
Version-1 authority data structure.

The following constant specifies the version number of the current version:

MQZAD_CURRENT_VERSION
Current version of authority data structure.

This is an input field to the service.

ProfileName (MQCHAR48):

Profile name.

For the *Filter* parameter, this field is the profile name whose authority data is required. If the name is entirely blank up to the end of the field or the first null character, authority data for all profile names is returned.

For the *AuthorityBuffer* parameter, this field is the name of a profile that matches the specified selection criteria.

ObjectType (MQLONG):

Object type.

For the *Filter* parameter, this field is the object type for which authority data is required. If the value is MQOT_ALL, authority data for all object types is returned.

For the *AuthorityBuffer* parameter, this field is the object type to which the profile identified by *ProfileName* applies.

The value is one of the following; for the *Filter* parameter, the value MQOT_ALL is also valid:

MQOT_AUTH_INFO
Authentication information.

MQOT_CHANNEL
Channel.

MQOT_CLNTCONN_CHANNEL
Client connection channel.

MQOT_LISTENER
Listener.

MQOT_NAMELIST
Namelist.

MQOT_PROCESS
Process definition.

MQOT_Q
Queue.

MQOT_Q_MGR
Queue manager.

MQOT_SERVICE
Service.

Authority (MQLONG):

Authority.

For the *Filter* parameter, this field is ignored.

For the *AuthorityBuffer* parameter, this field represents the authorizations that the entity has to the objects identified by *ProfileName* and *ObjectType*. If the entity has only one authority, the field is equal to the appropriate authorization value (MQZAO_* constant). If the entity has more than one authority, the field is the bitwise OR of the corresponding MQZAO_* constants.

EntityDataPtr (PMQZED):

Address of MQZED structure identifying an entity.

For the *Filter* parameter, this field points to an MQZED structure that identifies the entity whose authority data is required. If *EntityDataPtr* is the null pointer, authority data for all entities is returned.

For the *AuthorityBuffer* parameter, this field points to an MQZED structure that identifies the entity whose authority data has been returned.

EntityType (MQLONG):

Entity type.

For the *Filter* parameter, this field specifies the entity type for which authority data is required. If the value is MQZAET_NONE, authority data for all entity types is returned.

For the *AuthorityBuffer* parameter, this field specifies the type of the entity identified by the MQZED structure pointed to by *EntityDataPtr*.

The value is one of the following; for the *Filter* parameter, the value MQZAET_NONE is also valid:

MQZAET_PRINCIPAL
Principal.

MQZAET_GROUP
Group.

Options (MQAUTHOPT):

Options.

This field specifies options that give control over the profiles that are displayed.

One of the following must be specified:

MQAUTHOPT_NAME_ALL_MATCHING
Displays all profiles

MQAUTHOPT_NAME_EXPLICIT
Displays profiles that have exactly the same name as specified in the *ProfileName* field.

In addition, one of the following must also be specified:

MQAUTHOPT_ENTITY_SET
Display all profiles used to calculate the cumulative authority that the entity has to the object specified by *ProfileName*. The *ProfileName* field must not contain any wildcard characters.

- If the specified entity is a principal, for each member of the set {entity, groups} the most applicable profile that applies to the object is displayed.
- If the specified entity is a group, the most applicable profile from the group that applies to the object is displayed.
- If this value is specified, then the values of *ProfileName*, *ObjectType*, *EntityType*, and the entity name specified in the *EntityDataPtr* MQZED structure, must all be non-blank.

If you have specified *MQAUTHOPT_NAME_ALL_MATCHING*, you can also specify the following:

MQAUTHOPT_ENTITY_EXPLICIT
Displays profiles that have exactly the same entity name as the entity name specified in the *EntityDataPtr* MQZED structure.

C declaration

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQCHAR48  ProfileName;    /* Profile name */
  MQLONG    ObjectType;     /* Object type */
  MQLONG    Authority;      /* Authority */
  PMQZED    EntityDataPtr;  /* Address of MQZED structure identifying an
                               entity */
  MQLONG    EntityType;     /* Entity type */
  MQAUTHOPT Options;        /* Options */
};
```

MQZED – Entity descriptor

The MQZED structure describes the information that is passed to the MQZAS_VERSION_2 authorization service calls.

Fields

StrucId (MQCHAR4):

Structure identifier.

The value is:

MQZED_STRUC_ID
Identifier for entity descriptor structure.

For the C programming language, the constant MQZED_STRUC_ID_ARRAY is also defined; this has the same value as MQZED_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the service.

Version (MQLONG):

Structure version number.

The value is:

MQZED_VERSION_1
Version-1 entity descriptor structure.

The following constant specifies the version number of the current version:

MQZED_CURRENT_VERSION
Current version of entity descriptor structure.

This is an input field to the service.

EntityNamePtr (PMQCHAR):

Address of entity name.

This is a pointer to the name of the entity whose authorization is to be checked.

EntityDomainPtr (PMQCHAR):

Address of entity domain name.

This is a pointer to the name of the domain containing the definition of the entity whose authorization is to be checked.

SecurityId (MQBYTE40):

Security identifier.

This is the security identifier whose authorization is to be checked.

CorrelationPtr (MQPTR):

Correlation pointer.

This facilitates the passing of correlational data between the authenticate user function and other appropriate OAM functions.

C declaration

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  PMQCHAR   EntityNamePtr;    /* Address of entity name */
  PMQCHAR   EntityDomainPtr;  /* Address of entity domain name */
  MQBYTE40  SecurityId;       /* Security identifier */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
```

MQZFP – Free parameters

The MQZFP structure is used on the MQZ_FREE_USER call for the *FreeParms* parameter. This parameter specifies data related to resource to be freed.

Fields

StrucId (MQCHAR4):

Structure identifier.

The value is:

MQZFP_STRUC_ID
Identifier for free parameters structure.

For the C programming language, the constant MQZFP_STRUC_ID_ARRAY is also defined; this has the same value as MQZFP_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the service.

Version (MQLONG):

Structure version number.

The value is:

MQZFP_VERSION_1
Version-1 free parameters structure.

The following constant specifies the version number of the current version:

MQZFP_CURRENT_VERSION
Current version of free parameters structure.

This is an input field to the service.

Reserved (MQBYTE8):

Reserved field.

The initial value is null.

CorrelationPtr (MQPTR):

Correlation pointer.

Address of correlation data relating to the resource to be freed.

C declaration

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQBYTE8   Reserved;         /* Reserved field */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
};
```

MQZIC – Identity context

The MQZIC structure is used on the MQZ_AUTHENTICATE_USER call for the *IdentityContext* parameter.

The MQZIC structure contains identity context information, that identifies the user of the application that first put the message on a queue:

- The queue manager fills the UserIdentifier field with a name that identifies the user, the way that the queue manager can do this depends on the environment in which the application is running.
- The queue manager fills the AccountingToken field with a token or number that it determined from the application that put the message.
- Applications can use the ApplIdentityData field for any extra information that they want to include about the user (for example, an encrypted password).

Suitably authorized applications may set the identity context using the MQZ_AUTHENTICATE_USER function.

A Windows systems security identifier (SID) is stored in the AccountingToken field when a message is created under WebSphere MQ for Windows. The SID can be used to supplement the UserIdentifier field and to establish the credentials of a user.

Fields

StrucId (MQCHAR4):

Structure identifier.

The value is:

MQZIC_STRUC_ID
Identifier for identity context structure.

For the C programming language, the constant MQZIC_STRUC_ID_ARRAY is also defined; this has the same value as MQZIC_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the service.

Version (MQLONG):

Structure version number.

The value is:

MQZIC_VERSION_1
Version-1 identity context structure.

The following constant specifies the version number of the current version:

MQZIC_CURRENT_VERSION
Current version of identity context structure.

This is an input field to the service.

UserIdentifier (MQCHAR12):

User identifier.

This is part of the **identity context** of the message.

UserIdentifier specifies the user identifier of the application that originated the message. The queue manager treats this information as character data, but does not define the format of it. For more information on the *UserIdentifier* field, see WebSphere MQ Application Programming Reference.

AccountingToken (MQBYTE32):

Accounting token.

This is part of the **identity context** of the message.

AccountingToken allows an application to cause work done as a result of the message to be appropriately charged. The queue manager treats this information as a string of bits and does not check its content. For more information on the *AccountingToken* field, see WebSphere MQ Application Programming Reference.

ApplIdentityData (MQCHAR32):

Application data relating to identity.

This is part of the **identity context** of the message.

ApplIdentityData is information that is defined by the application suite that can be used to provide additional information about the origin of the message. For example, it could be set by applications running with suitable user authority to indicate whether the identity data is trusted. For more information on the *ApplIdentityData* field, see WebSphere MQ Application Programming Reference.

C declaration

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQCHAR12  UserIdentifier;   /* User identifier */
  MQBYTE32  AccountingToken;  /* Accounting token */
  MQCHAR32  ApplIdentityData; /* Application data relating to identity */
};
```

Chapter 11. The CL commands

This section provides reference information for the WebSphere MQ CL commands.

Connect MQ (CCTMQM)

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

The Connect Message Queue Manager (CCTMQM) command does not perform any function and is only provided for compatibility with previous releases of WebSphere MQ and MQSeries.

Parameters

None

Examples

None

Error messages

Unknown

Change Message Queue Manager (CHGMQM)

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

The Change Message Queue Manager (CHGMQM) command changes the specified attributes of the local queue manager.

Parameters

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

Keyword	Description	Choices	Notes
CLWLEXIT	Cluster workload exit	Single values: ***SAME** , *NONE Other values: *Qualified object name*	Optional, Positional 21
	Qualifier 1: Cluster workload exit	*Name*	
	Qualifier 2: Library	*Name*	
CLWLLEN	Cluster workload exit length	0-999999999, ***SAME**	Optional, Positional 22
REPOS	Repository name	*Character value*, *NONE, ***SAME**	Optional, Positional 23
REPOSNL	Repository name list	*Character value*, *NONE, ***SAME**	Optional, Positional 24
SSLCRLNL	SSL CRL Namelist	*Character value*, *NONE, ***SAME**	Optional, Positional 25
SSLKEYR	SSL Key Repository	*Character value*, *NONE, ***SAME** , *SYSTEM	Optional, Positional 26
SSLKEYRPWD	SSL Repository Password	*Character value*, *NONE, ***SAME**	Optional, Positional 27
SSLRSTCNT	SSL key reset count	0-999999999, ***SAME** , *NONE	Optional, Positional 28
IPADDRV	IP protocol	***SAME** , *IPV4, *IPV6	Optional, Positional 29
CLWLMRUC	Cluster workload channels	0-999999999, ***SAME**	Optional, Positional 30
CLWLUSEQ	Cluster workload queue use	***SAME** , *LOCAL, *ANY	Optional, Positional 31
LOGGEREVT	Log recovery events enabled	***SAME** , *YES, *NO	Optional, Positional 32
CHLEVT	Channel events enabled	***SAME** , *YES, *NO, *EXCEPTION	Optional, Positional 33
SSLEVT	SSL events enabled	***SAME** , *YES, *NO	Optional, Positional 34
SCHINIT	Channel initiator control	***SAME** , *QMGR, *MANUAL	Optional, Positional 35
SCMDSERV	Command server control	***SAME** , *QMGR, *MANUAL	Optional, Positional 36
MONQ	Queue Monitoring	***SAME** , *NONE, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 37
MONCHL	Channel Monitoring	***SAME** , *NONE, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 38
MONACLS	Cluster Sender Monitoring	***SAME** , *QMGR, *NONE, *LOW, *MEDIUM, *HIGH	Optional, Positional 39
STATMQI	Queue Manager Statistics	***SAME** , *OFF, *ON	Optional, Positional 40

Keyword	Description	Choices	Notes
STATQ	Queue Statistics	***SAME** , *NONE, *OFF, *ON	Optional, Positional 41
STATCHL	Channel Statistics	***SAME** , *NONE, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 42
STATACLS	Cluster Sender Statistics	***SAME** , *QMGR, *NONE, *LOW, *MEDIUM, *HIGH	Optional, Positional 43
STATINT	Statistics Interval	1-604800, ***SAME**	Optional, Positional 44
ACCTMQI	MQI Accounting	***SAME** , *OFF, *ON	Optional, Positional 45
ACCTQ	Queue Accounting	***SAME** , *NONE, *OFF, *ON	Optional, Positional 46
ACCTINT	Accounting Interval	1-604800, ***SAME**	Optional, Positional 47
ACCTCONO	Accounting Override	***SAME** , *ENABLED, *DISABLED	Optional, Positional 48
ROUTEREC	Trace Route Recording	***SAME** , *MSG, *QUEUE, *DISABLED	Optional, Positional 49
ACTIVREC	Activity Recording	***SAME** , *MSG, *QUEUE, *DISABLED	Optional, Positional 50
MAXPROPLEN	Maximum Property Data Length	0-104857600, ***SAME** , *ANY	Optional, Positional 51
MARKINT	Message mark-browse interval	0-999999999, ***SAME** , *ANY	Optional, Positional 52
PSRTYCNT	PubSub max msg retry count	0-999999999, ***SAME**	Optional, Positional 53
PSNPMMSG	PubSub NPM msg	***SAME** , *DISCARD, *KEEP	Optional, Positional 54
PSNPMRES	PubSub NPM msg response	***SAME** , *NORMAL, *SAFE, *DISCARD, *KEEP	Optional, Positional 55
PSSYNCPT	PubSub syncpoint	***SAME** , *YES, *IFPER	Optional, Positional 56
PSMODE	Pubsub Engine Control	***SAME** , *ENABLED, *DISABLED, *COMPATIBLE	Optional, Positional 57

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Force (FORCE)

Specifies whether the command should be forced to complete if both of the following are true:

- DFTTMQ is specified.
- An application has a remote queue open, the resolution of which will be affected by this change.

The possible values are:

***NO** The command fails if an open remote queue will be affected.

***YES** The command is forced to complete.

Text 'description' (TEXT)

Specifies the text that briefly describes the queue manager definition.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Trigger interval (TRGITV)

Specifies the trigger time interval, expressed in milliseconds, to be used with queues that have TRGTYPE(*FIRST) specified.

When TRGTYPE(*FIRST) is specified the arrival of a message on a previously empty queue causes a trigger message to be generated. Any further messages that arrive on the queue within the specified interval will not cause a further trigger message to be generated.

The possible values are:

***SAME**
The attribute is unchanged.

interval-value
Specify a value in the range 0 through 999999999.

Undelivered message queue (UDLMSGQ)

Specifies the name of the local queue that is to be used for undelivered messages. Messages are put on this queue if they cannot be routed to their correct destination.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
There is no undelivered-message queue. The attribute is set to a blank string.

undelivered-message-queue-name
Specify the name of a local queue that is to be used as the undelivered-message queue.

Default transmission queue (DFTTMQ)

Specifies the name of the local transmission queue that is to be used as the default transmission queue. Messages transmitted to a remote queue manager are put on the default transmission queue if there is no transmission queue defined for their destination.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
There is no default transmission queue. The attribute is set to a blank string.

default-transmission-queue-name
Specify the name of a local transmission queue that is to be used as the default transmission queue.

Maximum handle limit (MAXHDL)

Specifies the maximum number of handles that any one job can have open at the same time.

The possible values are:

***SAME**
The attribute is unchanged.

maximum-handle-limit
Specify a value in the range 0 through 999999999.

Maximum uncommitted messages (MAXUMSG)

Specifies the maximum number of uncommitted messages. That is:

- The number of messages that can be retrieved, plus
- The number of messages that can be put, plus
- Any trigger and report messages generated within this unit of work, under any one syncpoint.

This limit does not apply to messages that are retrieved or put outside syncpoint.

The possible values are:

***SAME**
The attribute is unchanged.

maximum-uncommitted-messages
Specify a value in the range 1 through 999999999.

Authorization events enabled (AUTEVT)

Specifies whether authorization (Not Authorized) events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Authorization events are not generated.

***YES** Authorization events are generated.

Inhibit events enabled (INHEVT)

Specifies whether inhibit events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Inhibit events are not generated.

***YES** Inhibit events are generated.

Local error events enabled (LCLERREVT)

Specifies whether local error events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Local error events are not generated.

***YES** Local error events are generated.

Remote error events enabled (RMTERREVT)

Specifies whether remote error events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Remote error events are not generated.

***YES** Remote error events are generated.

Performance events enabled (PFREVT)

Specifies whether performance events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Performance events are not generated.

***YES** Performance events are generated.

Start and stop events enabled (STRSTPEVT)

Specifies whether start and stop events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Start and stop events are not generated.

***YES** Start and stop events are generated.

Automatic Channel Definition (CHAD)

Specifies whether receiver and server-connection channels are automatically defined.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Receiver and server-connection channels are not automatically defined.

***YES** Receiver and server-connection channels are automatically defined.

Auto Chan. Def. events enabled (CHADEV)

Specifies whether automatic channel definition events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Automatic channel definition events are not generated.

***YES** Automatic channel definition events are generated.

Auto Chan. Def. exit program (CHADEXIT)

Specifies the entry point of the program to be called as the automatic channel-definition exit.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No automatic channel definition exit is invoked.

channel-definition-exit-name
Specify the name of the channel definition exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified and the values *LIBL and *CURLIB are not permitted.

Maximum Message Length (MAXMSGL)

Specifies the maximum message length of messages (in bytes) allowed on queues for this queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

maximum-message-length
Specify a value in bytes, in the range 32 KB through 100 MB.

Coded Character Set (CCSID)

The coded character set identifier for the queue manager.

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

If you use this keyword to change the CCSID, applications that are running when the change is applied continue to use the original CCSID. You must stop and restart all running applications before you continue. This includes the command server and channel programs. You are recommended to stop and restart the queue manager after making the change to achieve this.

The possible values are:

***SAME**
The attribute is unchanged.

number
Specify a value in the range 1 through 65535. The value must represent a coded character set identifier (CCSID) that is recognised by the system.

Cluster Workload Exit Data (CLWLDATA)

Specifies the cluster workload exit data (maximum length 32 characters).

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The cluster workload exit data is not specified.

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

Cluster Workload Exit (CLWLEXIT)

Specifies the entry point of the program to be called as the cluster-workload exit.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No cluster-workload exit is invoked.

cluster-workload-exit
You must specify a fully-qualified name, when you specify a cluster-workload exit. In this instance, the libraries defined as *LIBL and *CURLIB are not permitted.

Cluster Workload Exit Data Length (CLWLLEN)

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

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-exit-data-length
Specify a value in bytes, in the range 0 through 999999999.

Repository name (REPOS)

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

If the parameter REPOSNL is non-blank this parameter must be blank.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
A cluster is not specified.

clustername
The maximum length is 48 characters conforming to the rules for naming WebSphere MQ objects.

Repository name list (REPOSNL)

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

If the parameter REPOS is non-blank this parameter must be blank.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
A namelist of clusters is not specified.

namelist
The name of the namelist.

SSL CRL Namelist (SSLCRLNL)

The name of a namelist of authinfo objects which this queue manager uses to check certificate status.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
A namelist of authinfo objects is not specified.

namelist
The name of the namelist.

SSL Key Repository (SSLKEYR)

The location of a key repository for this queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

***SYSTEM**
The queue manager uses the *SYSTEM key repository. Setting the SSLKEYR repository to this value causes the queue manager to be registered as an application to Digital Certificate Manager. You can assign any client or server certificate in the *SYSTEM store to the queue manager through Digital Certificate Manager. If you specify this value you are not required to set the key repository password (SSLKEYRPWD).

***NONE**
A key repository is not specified.

filename
The location of the key repository. If you specify this value you are must ensure the key repository contains a correctly labelled digital certificate and also set the key repository password (SSLKEYRPWD) to enable channels to access the key repository. See the WebSphere MQ Security manual for more details.

SSL Repository Password (SSLKEYRPWD)

The password of a key repository for this queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
A key repository password is not specified.

password
The password of the repository.

SSL key reset count (SSLRSTCNT)

Specifies when SSL channel MCAs that initiate communication reset the secret key used for encryption on the channel. The value represents the total number of unencrypted bytes that are sent and received on the channel before the secret key is renegotiated. The number of bytes includes control information sent by the message channel agent.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Secret key renegotiation is disabled.

key-reset-byte-count
Specify a value in bytes, in the range 0 through 999999999. A value of 0 indicates that secret key renegotiation is disabled.

IP protocol (IPADDRV)

The IP protocol to use for channel connections.

This attribute is only relevant for systems enabled for both IPv4 and IPv6. The attribute affects channels with TRPTYPE defined as TCP when the CONNAME is defined as a hostname that resolves to both an IPv4, and an IPv6 address, and one of the following is true:

- LOCLADDR is not specified.
- LOCLADDR also resolves to both an IPv4 and an IPv6 address.

The possible values are:

***SAME**
The attribute is unchanged.

***IPV4** The IPv4 stack is used.

***IPV6** The IPv6 stack is used.

Cluster workload channels (CLWLMRUC)

Specifies the maximum number of most-recently-used cluster channels, to be considered for use by the cluster workload choice algorithm.

The possible values are:

***SAME**
The attribute is unchanged.

maximum-cluster-workload-channels
Specify a value in the range 0 through 999999999.

Cluster workload queue use (CLWLUSEQ)

Specifies the behaviour of an MQPUT when the target queue has both a local instance and at least one remote cluster instance. If the put originates from a cluster channel then this attribute does not apply. This value is used for queues whose CLWLUSEQ value is *QMGR.

The possible values are:

***SAME**
The attribute is unchanged.

***LOCAL**
The local queue will be the sole target of the MQPUT.

***ANY** The queue manager will treat such a local queue as another instance of the cluster queue for the purposes of workload distribution.

Log recovery events enabled (LOGGEREVT)

Specifies whether log recovery events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Log recovery events are not generated.

***YES** Log recovery events are generated.

Channel events enabled (CHLEVT)

Specifies whether channel events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Channel events are not generated.

***EXCEPTION**
Exception channel events are generated.

Only the following channel events are generated:

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- MQRC_CHANNEL_STOPPED

  The channel events are issued with the following reason qualifiers:
  - MQRQ_CHANNEL_STOPPED_ERROR
  - MQRQ_CHANNEL_STOPPED_RETRY
  - MQRQ_CHANNEL_STOPPED_DISABLED
  - MQRC_CHANNEL_STOPPED_BY_USER

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

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

- MQRC_CHANNEL_STARTED
- MQRC_CHANNEL_STOPPED

  with the following reason qualifier:
  - MQRQ_CHANNEL_STOPPED_OK

SSL events enabled (SSLEVT)

Specifies whether SSL events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

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

***YES** SSL events are generated.

The following event is generated:

- MQRC_CHANNEL_SSL_ERROR

Channel initiator control (SCHINIT)

Specifies the channel initiator control.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
Start and stop the channel initiator with the queue manager.

***MANUAL**
Do not automatically start the channel initiator with the queue manager.

Command server control (SCMDSERV)

Specifies the command server control.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
Start and stop the command server with the queue manager.

***MANUAL**
Do not automatically start the command server with the queue manager.

Queue Monitoring (MONQ)

Controls the collection of online monitoring data for queues.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Online monitoring data for queues is disabled regardless of the setting of the MONQ queue attribute.

***OFF** Monitoring data collection is turned off for queues specifying *QMGR in the MONQ queue attribute.

***LOW** Monitoring data collection is turned on with a low ratio of data collection for queues specifying *QMGR in the MONQ queue attribute.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection for queues specifying *QMGR in the MONQ queue attribute.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection for queues specifying *QMGR in the MONQ queue attribute.

Channel Monitoring (MONCHL)

Controls the collection of online monitoring data for channels.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Online monitoring data for channels is disabled regardless of the setting of the MONCHL channel attribute.

***OFF** Monitoring data collection is turned off for channels specifying 'QMGR' in the MONCHL queue attribute.

***LOW** Monitoring data collection is turned on with a low ratio of data collection for channels specifying *QMGR in the MONCHL channel attribute.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection for channels specifying *QMGR in the MONCHL channel attribute.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection for channels specifying *QMGR in the MONCHL channel attribute.

Cluster Sender Monitoring (MONACLS)

Controls the collection of online monitoring data for auto-defined cluster sender channels. The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Online monitoring data for auto-defined cluster sender channels is disabled.

***QMGR**
The collection of Online Monitoring Data is inherited from the setting of the MONCHL attribute in the QMGR object.

***LOW** Monitoring data collection is turned on with a low ratio of data collection for auto-defined cluster sender channels.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection for auto-defined cluster sender channels.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection for auto-defined cluster sender channels.

Queue Manager Statistics (STATMQI)

Controls the collection of statistics monitoring information for the queue manager. The possible values are:

***SAME**
The attribute is unchanged.

***OFF** Data collection for MQI statistics is disabled.

***ON** Data collection for MQI statistics is enabled.

Queue Statistics (STATQ)

Controls the collection of statistics data for queues. The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Data collection for queue statistics is disabled for all queues regardless of the setting of the STATQ queue attribute.

***OFF** Statistics data collection is turned off for queues specifying *QMGR in the STATQ queue attribute.

***ON** Statistics data collection is turned on for queues specifying *QMGR in the STATQ queue attribute.

Channel Statistics (STATCHL)

Controls the collection of statistics data for channels. The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Data collection for channel statistics is disabled for all channels regardless of the setting of the STATCHL channel attribute.

***OFF** Statistics data collection is turned off for channels specifying *QMGR in the STATCHL channel attribute.

***LOW** Statistics data collection is turned on with a low ratio of data collection for channels specifying *QMGR in the STATCHL channel attribute.

***MEDIUM**
Statistics data collection is turned on with a moderate ratio of data collection for channels specifying *QMGR in the STATCHL channel attribute.

***HIGH**
Statistics data collection is turned on with a high ratio of data collection for channels specifying *QMGR in the STATCHL channel attribute.

Cluster Sender Statistics (STATACLS)

Controls the collection of statistics data for auto-defined cluster sender channels. The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Statistics data collection for auto-defined cluster sender channels is disabled.

***LOW** Statistics data collection for auto-defined cluster sender channels is enabled with a low ratio of data collection.

***MEDIUM**
Statistics data collection for auto-defined cluster sender channels is enabled with a moderate ratio of data collection.

***HIGH**
Statistics data collection for auto-defined cluster sender channels is enabled with a high ratio of data collection.

Statistics Interval (STATINT)

How often (in seconds) statistics monitoring data is written to the monitoring Queue.

The possible values are:

***SAME**
The attribute is unchanged.

statistics-interval
Specify a value in the range 1 through 604800.

MQI Accounting (ACCTMQI)

Controls the collection of accounting information for MQI data. The possible values are:

***SAME**
The attribute is unchanged.

***OFF** API accounting data collection is switched off.

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

Queue Accounting (ACCTQ)

Controls the collection of accounting information for queues. The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Accounting data collection for queues is disabled and may not be overridden using the queue attribute ACCTQ.

***OFF** Accounting data collection is turned off for queues specifying *QMGR in the ACCTQ queue attribute.

***ON** Accounting data collection is turned on for queues specifying *QMGR in the ACCTQ queue attribute.

Accounting Interval (ACCTINT)

After how long in seconds, intermediate accounting records are written.

The possible values are:

***SAME**
The attribute is unchanged.

accounting-interval
Specify a value in the range 1 through 604800.

Accounting Override (ACCTCONO)

Whether applications can override the setting of the ACCTMQI and the ACCTQ values in the QMGR attribute. The possible values are:

***SAME**
The attribute is unchanged.

***ENABLED**
Application may override the setting of the ACCTMQI and ACCTQ QMGR attributes using the Options field in the MQCNO structure on the MQCONNX api call.

***DISABLED**
Application may not override the setting of the ACCTMQI and ACCTQ QMGR attributes using the Options field in the MQCNO structure on the MQCONNX api call.

Trace Route Recording (ROUTEREC)

Controls the recording of trace route information.

The possible values are:

***SAME**
The attribute is unchanged.

***MSG** Reply put to destination specified by the message.

***QUEUE**
Reply put to fixed name queue.

***DISABLED**
No appending to trace route messages allowed.

Activity Recording (ACTIVREC)

Controls the generation of activity reports.

The possible values are:

***SAME**
The attribute is unchanged.

***MSG** Report put to destination specified by the message.

***QUEUE**
Report put to fixed name queue.

***DISABLED**
No activity reports are generated.

Maximum Property Data Length (MAXPROPLEN)

Specifies a maximum length for property data.

The possible values are:

***SAME**
The attribute is unchanged.

***ANY** There is no limit on the length of property data.

max-property-data-length
Specify a value in bytes, in the range 0 through 104857600 (ie: 10 MB).

Message mark-browse interval (MARKINT)

An approximate time interval in milliseconds, for which messages that have been marked-browsed by a call to MQGET with the get message option MQGMO_MARK_BROWSE_CO_OP are expected to remain marked-browsed.

The possible values are:

***SAME**
The attribute is unchanged.

***ANY** Messages will remain marked-browsed indefinitely.

A time interval
A time interval expressed in milliseconds, in the range 0 through 999999999.

PubSub max msg retry count (PSRTYCNT)

The number of retries when processing (under syncpoint) a failed command message.

The possible values are:

***SAME**
The attribute is unchanged.

Retry count
Specify a value in the range 0 through 999999999.

PubSub NPM msg (PSNPMMSG)

Whether to discard (or keep) a undelivered input message

The possible values are:

***SAME**
The attribute is unchanged.

***DISCARD**
Non-persistent input messages may be discarded if they cannot be processed.

***KEEP**
Non-persistent input messages will not be discarded if they cannot be processed. In this situation the QPUBSUB will continue to retry the process at appropriate intervals and does not continue processing subsequent messages.

PubSub NPM msg response (PSNPMRES)

Controls the behavior of undelivered response messages

The possible values are:

***SAME**
The attribute is unchanged.

***NORMAL**
Non-persistent responses which cannot be placed on the reply queue are put on the dead letter queue, if they cannot be placed on the DLQ then they are discarded.

***SAFE** Non-persistent responses which cannot be placed on the reply queue are put on the dead letter queue. If the response cannot be set and cannot be placed on the DLQ then the QPUBSUB will roll back the current operation and then retry at appropriate intervals and does not continue processing subsequent messages.

***DISCARD**
Non-persistent responses are not placed on the reply queue are discarded.

***KEEP**
Non-persistent responses are not placed on the dead letter queue or discarded. Instead, the QPUBSUB will back out the current operation and then retry it at appropriate intervals.

PubSub syncpoint (PSSYNCPT)

Whether only persistent (or all) messages should be processed under syncpoint

The possible values are:

***SAME**
The attribute is unchanged.

***IFPER**
This makes the queued pubsub daemon receive non-persistent messages outside syncpoint. If the daemon receives a publication outside syncpoint, the daemon forwards the publication to subscribers known to it outside syncpoint.

***YES** This makes the queued pubsub daemon receive all messages under syncpoint.

Pubsub Engine Control (PSMODE)

Pubsub Engine Control.

The possible values are:

***SAME**
The attribute is unchanged.

***ENABLED**
The publish/subscribe engine and the queued publish/subscribe interface are running. It is therefore possible to publish/subscribe by using the application programming interface, the queues that are being monitored by the queued publish/subscribe interface, or both.

***DISABLED**
The publish/subscribe engine and the queued publish/subscribe interface are not running. It is not possible to publish/subscribe by using the application programming interface. Any publish/subscribe messages put to the queues that are monitored by the queued publish/subscribe interface will not be acted upon.

***COMPATIBLE**
The publish/subscribe engine is running. It is possible to publish subscribe by using the application programming interface. The queued publish/subscribe interface is not running. Any publish/subscribe messages put to the queues that are monitored by the queued publish/subscribe interface will not be acted upon. Use this for compatibility with Websphere Business Integration Message Broker V6, or earlier versions, using this queue manager, because Websphere Business

Integration Message Broker needs to read the same queues from which the queued publish/subscribe interface would normally read.

Examples

None

Error messages

Unknown

Change MQ AuthInfo object (CHGMQMAUTI)

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

The Change MQ AuthInfo object (CHGMQMAUTI) command changes the specified attributes of an existing MQ authentication information object.

Parameters

Keyword	Description	Choices	Notes
AINAME	AuthInfo name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
CONNAME	Connection name	*Character value*, ***SAME**	Optional, Positional 3
TEXT	Text 'description'	*Character value*, ***SAME**, *NONE	Optional, Positional 4
USERNAME	User name	*Character value*, ***SAME**, *NONE	Optional, Positional 5
PASSWORD	User password	*Character value*, ***SAME**, *NONE	Optional, Positional 6

AuthInfo name (AINAME)

The name of the authentication information object to change.

The possible values are:

authentication-information-name
Specify the name of the authentication information object. The maximum string length is 48 characters.

Message Queue Manager name (MQMNAME)

The name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of an existing message queue manager. The maximum string length is 48 characters.

Connection name (CONNAME)

The DNS name or IP address of the host on which the LDAP server is running, together with an optional port number. The default port number is 389. No default is provided for the DNS name or IP address.

The possible values are:

***SAME**
The connection name remains unchanged from the original authentication information object.

connection-name
Specify the fully qualified DNS name or IP address of the host together with an optional port number. The maximum string length is 264 characters.

Text 'description' (TEXT)

A short text description of the authentication information object.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SAME**
The text string is unchanged.

***NONE**
The text is set to a blank string.

description
The string length can be up to 64 characters enclosed in apostrophes.

User name (USERNAME)

The distinguished name of the user that is binding to the directory. The default user name is blank.

The possible values are:

***SAME**
The user name is unchanged.

***NONE**
The user name is blank.

LDAP-user-name
Specify the distinguished name of the LDAP user. The maximum string length is 1024 characters.

User password (PASSWORD)

The password for the LDAP user.

The possible values are:

***SAME**
The password is unchanged.

***NONE**
The password is blank.

LDAP-password
The LDAP user password. The maximum string length is 32 characters.

Examples

None

Error messages

Unknown

Change MQ Channel (CHGMQMCHL)

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

The Change MQ Channel (CHGMQMCHL) command changes the specified attributes of an existing MQ channel definition.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN	Optional, Key, Positional 3
TRPTYPE	Transport type	*LU62, *TCP, ***SAME**	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
TGTMQMNAME	Target Queue Manager	*Character value*, *NONE, ***SAME**	Optional, Positional 6
CONNAME	Connection name	*Character value*, *NONE, ***SAME**	Optional, Positional 7
TPNAME	Transaction Program Name	*Character value*, *BLANK, ***SAME**	Optional, Positional 8
MODENAME	Mode Name	*Character value*, *BLANK, ***SAME**	Optional, Positional 9
TMQNAME	Transmission queue	*Character value*, ***SAME**	Optional, Positional 10
MCANAME	Message channel agent	Single values: ***SAME** , *NONE Other values: *Qualified object name*	Optional, Positional 11
	Qualifier 1: Message channel agent	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MCAUSRID	Message channel agent user ID	*Character value*, *NONE, *PUBLIC, ***SAME**	Optional, Positional 12

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

Keyword	Description	Choices	Notes
RCVEXIT	Receive exit	Single values: ***SAME** , *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 27
	Qualifier 1: Receive exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CRCVEXIT	Receive exit	Single values: ***SAME** , *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 28
RCVUSRDATA	Receive exit user data	Values (up to 10 repetitions): *Character value*, ***SAME** , *NONE	Optional, Positional 29
MSGEXIT	Message exit	Single values: ***SAME** , *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 30
	Qualifier 1: Message exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGUSRDATA	Message exit user data	Values (up to 10 repetitions): *Character value*, ***SAME** , *NONE	Optional, Positional 31
MSGRTYEXIT	Message retry exit	Single values: ***SAME** , *NONE Other values: *Qualified object name*	Optional, Positional 32
	Qualifier 1: Message retry exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGRTYDATA	Message retry exit data	*Character value*, ***SAME** , *NONE	Optional, Positional 33
MSGRTYNBR	Number of message retries	0-999999999, ***SAME**	Optional, Positional 34
MSGRTYITV	Message retry interval	0-999999999, ***SAME**	Optional, Positional 35
CVTMSG	Convert message	*YES, *NO, ***SAME**	Optional, Positional 36
PUTAUT	Put authority	*DFT, *CTX, ***SAME**	Optional, Positional 37
SEQNUMWRAP	Sequence number wrap	100-999999999, ***SAME**	Optional, Positional 38
MAXMSGLEN	Maximum message length	0-104857600, ***SAME**	Optional, Positional 39

Keyword	Description	Choices	Notes
HRTBTINTVL	Heartbeat interval	0-999999999, ***SAME**	Optional, Positional 40
NPMSPEED	Non Persistent Message Speed	*FAST, *NORMAL, ***SAME**	Optional, Positional 41
CLUSTER	Cluster Name	*Character value*, *NONE, ***SAME**	Optional, Positional 42
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SAME**	Optional, Positional 43
NETPRTY	Network Connection Priority	0-9, ***SAME**	Optional, Positional 44
SSLCIPH	SSL CipherSpec	*Character value*, *NULL_MD5, *NULL_SHA, *RC4_MD5_EXPORT, *RC4_MD5_US, *RC4_SHA_US, *RC2_MD5_EXPORT, *DES_SHA_EXPORT, *TRIPLE_DES_SHA_US, *AES_SHA_US, *TLS_RSA_WITH_NULL_MD5, *TLS_RSA_WITH_NULL_SHA, *TLS_RSA_EXPORT_WITH_RC4_40_MD5, *TLS_RSA_WITH_RC4_128_MD5, *TLS_RSA_WITH_RC4_128_SHA, *TLS_RSA_EXPORT_WITH_RC2_40_MD5, *TLS_RSA_WITH_DES_CBC_SHA, *TLS_RSA_WITH_3DES_EDE_CBC_SHA, *TLS_RSA_WITH_AES_128_CBC_SHA, *TLS_RSA_WITH_AES_256_CBC_SHA, *NONE, ***SAME**	Optional, Positional 45
SSLCAUTH	SSL Client Authentication	*REQUIRED, *OPTIONAL, ***SAME**	Optional, Positional 46
SSLPEER	SSL Peer name	*Character value*, *NONE, ***SAME**	Optional, Positional 47
LOCLADDR	Local communication address	*Character value*, *NONE, ***SAME**	Optional, Positional 48
BATCHHB	Batch Heartbeat Interval	0-999999999, ***SAME**	Optional, Positional 49
USERID	Task user identifier	*Character value*, *NONE, ***SAME**	Optional, Positional 50
PASSWORD	Password	*Character value*, *NONE, ***SAME**	Optional, Positional 51
KAINT	Keep Alive Interval	0-99999, ***SAME** , *AUTO	Optional, Positional 52

Keyword	Description	Choices	Notes
COMPHDR	Header Compression	Values (up to 2 repetitions): *NONE, *SYSTEM, ***SAME**	Optional, Positional 53
COMPMSG	Message Compression	Single values: *ANY Other values (up to 4 repetitions): *NONE, *RLE, *ZLIBHIGH, *ZLIBFAST, ***SAME**	Optional, Positional 54
MONCHL	Channel Monitoring	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SAME**	Optional, Positional 55
STATCHL	Channel Statistics	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SAME**	Optional, Positional 56
CLWLRANK	Cluster Workload Rank	0-9, ***SAME**	Optional, Positional 57
CLWLPRTY	Cluster Workload Priority	0-9, ***SAME**	Optional, Positional 58
CLWLWGHT	Cluster Channel Weight	1-99, ***SAME**	Optional, Positional 59
SHARECNV	Sharing Conversations	0-999999999, ***SAME**	Optional, Positional 60
PROPCTL	Property Control	*COMPAT, *NONE, *ALL, ***SAME**	Optional, Positional 61
MAXINST	Maximum Instances	0-999999999, ***SAME**	Optional, Positional 62
MAXINSTC	Maximum Instances Per Client	0-999999999, ***SAME**	Optional, Positional 63
CLNTWGHT	Client Channel Weight	0-99, ***SAME**	Optional, Positional 64
AFFINITY	Client Channel Affinity	*PREFERRED, *NONE, ***SAME**	Optional, Positional 65

Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

channel-name
Specify the channel name.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Channel type (CHLTYPE)

Specifies the type of the channel being changed.

The possible values are:

***SDR** Sender channel

***SVR** Server channel

***RCVR**
Receiver channel

***RQSTR**
Requester channel

***SVRCN**
Server-connection channel

***CLUSSDR**
Cluster-sender channel

***CLUSRCVR**
Cluster-receiver channel

***CLTCN**
Client-connection channel

Transport type (TRPTYPE)

Specifies the transmission protocol.

The possible values are:

***SAME**
The attribute is unchanged.

***LU62** SNA LU 6.2.

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

Text 'description' (TEXT)

Specifies text that briefly describes the channel definition.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Target Queue Manager (TGTMQMNAME)

Specifies the name of the target queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The name of the target queue manager for a client connection channel (CHLTYPE) *CLTCN is unspecified.

message-queue-manager-name
The name of the target message queue manager for a client connection channel (CHLTYPE) *CLTCN.

For other channel types this parameter must not be specified.

Connection name (CONNAME)

Specifies the name of the machine to connect.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The connection name is blank.

connection-name
Specify the connection name as required by the transmission protocol:

- For *LU62, specify the name of the CSI object.

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

  If a connection name is not specified for cluster-receiver channels a connection name is automatically generated, assuming the default port and using the current IP address of the system.

  Where a port is not specified the default port 1414 is assumed.

For cluster-receiver channels the connection name relates to the local queue manager, and for other channels it relates to the target queue manager.

This parameter is required for channels with channel type (CHLTYPE) of *SDR, *RQSTR, *CLTCN and *CLUSSDR. It is optional for *SVR and *CLUSRCVR channels, and is not valid for *RCVR or *SVRCN channels.

Transaction Program Name (TPNAME)

This parameter is valid for channels with a TRPTYPE defined as LU 6.2 only.

This parameter must be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it must be set to blanks. The name is taken instead from the CPI-C Communications Side Object.

This parameter is not valid for channels with a CHLTYPE defined as *RCVR.

The possible values are:

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

***NONE**
No transaction program name is specified.

***BLANK**
The transaction program name is taken from CPI-C Communications Side Object. The side object name must be specified in the CONNAME parameter.

transaction-program-name
Specify the SNA transaction program name.

Mode Name (MODENAME)

This parameter is valid for channels with a TRPTYPE defined as LU 6.2. If TRPTYPE is not defined as LU 6.2 the data is ignored and no error message is issued.

If specified, the value must be set to the SNA mode name, unless the CONNAME contains a side-object name, in which case it must be set to blanks. The name is then taken from the CPI-C Communications Side Object.

This parameter is not valid for channels with CHLTYPE defined as *RCVR or *SVRCONN.

The possible values are:

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

***NONE**
No mode name is specified.

***BLANK**
Name will be taken from the CPI-C Communications Side Object. This must be specified in the CONNAME parameter.

SNA-mode-name
Specify the SNA Mode Name

Transmission queue (TMQNAME)

Specifies the name of the transmission queue.

The possible values are:

***SAME**
The attribute is unchanged.

transmission-queue-name
Specify the name of the transmission queue. A transmission queue name is required if the CHLTYPE is defined as *SDR or *SVR.

For other channel types this parameter must not be specified.

Message channel agent (MCANAME)

This parameter is reserved and should not be used.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The MCA program name is blank.

This parameter cannot be specified if the CHLTYPE is defined as *RCVR, *SVRCN, or *CLTCN.

Message channel agent user ID (MCAUSRID)

Specifies the message channel agent user identifier which is to be used by the message channel agent for authorization to access MQ resources, including (if PUTAUT is *DFT) authorization to put the message to the destination queue for receiver or requester channels.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The message channel agent uses its default user identifier.

***PUBLIC**
Uses the public authority.

mca-user-identifier
Specify the user identifier to be used.

This parameter cannot be specified for a channel type (CHLTYPE) of *CLTCN.

Message channel agent Type (MCATYPE)

Specifies whether the message channel agent program should run as a thread or as a process.

The possible values are:

***SAME**
The attribute is unchanged.

***PROCESS**
The message channel agent runs as a separate process.

***THREAD**
The message channel agent runs as a separate thread.

This parameter can only be specified for channels with CHLTYPE defined as *SDR, *SVR, *RQSTR, *CLUSSDR or *CLUSRCVR.

Batch Interval (BATCHINT)

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

The batch is terminated by which ever of the following occurs first: BATCHSZ messages have been sent, or the transmission queue is empty and BATCHINT is exceeded.

The default value is 0, which means that the batch is terminated as soon as the transmission queue becomes empty (or the BATCHSZ limit is reached).

The value must be in the range 0 through 999999999.

This parameter is valid for channels with CHLTYPE defined as *SDR, *SVR, *CLUSSDR, or *CLUSRCVR.

The possible values are:

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

batch-interval
Specify a value ranging from 0 through 999999999

Batch size (BATCHSIZE)

Specifies the maximum number of messages that can be sent down a channel before a checkpoint is taken.

The possible values are:

***SAME**
The attribute is unchanged.

batch-size
Specify a value ranging from 1 through 9999.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Disconnect interval (DSCITV)

Specifies the disconnect interval, which defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before closing the channel.

The possible values are:

***SAME**
The attribute is unchanged.

disconnect-interval
Specify a value ranging from 0 through 999999.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR or *CLTCN.

Short retry interval (SHORTTMR)

Specifies the short retry wait interval for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. This defines the interval between attempts to establish a connection to the remote machine.

The possible values are:

***SAME**
The attribute is unchanged.

short-retry-interval
Specify a value ranging from 0 through 999999999.

Short retry count (SHORTRTY)

Specifies the short retry count for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of attempts that are made to establish a connection to the remote machine, at intervals specified by SHORTTMR, before the (normally longer) LONGRTY and LONGTMR are used.

The possible values are:

***SAME**
The attribute is unchanged.

short-retry-count
Specify a value ranging from 0 through 999999999. A value of 0 means that no retries are allowed.

Long retry interval (LONGTMR)

Specifies the long retry wait interval for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by SHORTRTY has been exhausted.

The possible values are:

***SAME**
The attribute is unchanged.

long-retry-interval
Specify a value in the range 0 through 999999999.

Note: For implementation reasons, the maximum retry interval that can be used is 999999; values exceeding this are treated as 999999.

Long retry count (LONGRTY)

Specifies the long retry count for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of further attempts that are made to connect to the remote machine, at intervals specified by LONGTMR, after the count specified by SHORTRTY has been exhausted. An error message is logged if the connection is not established after the defined number of attempts.

The possible values are:

***SAME**
The attribute is unchanged.

long-retry-count
Specify a value in the range 0 through 999999999. A value of 0 means that no retries are allowed.

Security exit (SCYEXIT)

Specifies the name of the program to be called as the security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

  Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.
- On receipt of a response to a security message flow.

  Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The security exit program is not invoked.

security-exit-name
Specify the name of the security exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

Security exit (CSCYEXIT)

Specifies the name of the program to be called as the client security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

  Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.
- On receipt of a response to a security message flow.

  Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The client security exit program is not invoked.

security-exit-name
Specify the name of the client security exit program.

Security exit user data (SCYUSRDATA)

Specifies a maximum of 32 characters of user data that is passed to the security exit program.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data for the security exit program is not specified.

security-exit-user-data
Specify the user data for the security exit.

Send exit (SNDEXIT)

Specifies the entry point of the program to be called as the send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The send exit program is not invoked.

send-exit-name
Specify the name of the send exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

Send exit (CSNDEXIT)

Specifies the entry point of the program to be called as the client send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The client send exit program is not invoked.

send-exit-name
Specify the name of the client send exit program.

Send exit user data (SNDUSRDATA)

Specifies a maximum of 32 characters of user data that is passed to the send exit program.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data for the send exit program is not specified.

send-exit-user-data
Specify the user data for the send exit program.

Receive exit (CRCVEXIT)

Specifies the entry point of the program to be called as the client receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The client receive exit program is not invoked.

receive-exit-name
Specify the name of the client receive exit program.

Receive exit (RCVEXIT)

Specifies the entry point of the program to be called as the receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The receive exit program is not invoked.

receive-exit-name
Specify the name of the receive exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

Receive exit user data (RCVUSRDATA)

Specifies a maximum of 32 characters of user data that is passed to the receive exit program.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data for the receive exit program is not specified.

receive-exit-user-data
Specify a maximum of 32 characters of user data for the receive exit.

Message exit (MSGEXIT)

Specifies the entry point of the program to be called as the message exit. If a nonblank name is defined, the exit is invoked immediately after a message has been retrieved from the transmission queue. The exit is given the entire application message and message descriptor for modification.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The message exit program is not invoked.

message-exit-name
Specify the name of the message exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Message exit user data (MSGUSRDATA)

Specifies user data that is passed to the message exit program.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data for the message exit program is not specified.

message-exit-user-data
Specify a maximum of 32 characters of user data that is passed to the message exit program.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Message retry exit (MSGRTYEXIT)

Specifies the entry point of the program to be called as the message retry exit.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The message retry exit program is not invoked.

message-retry-exit-name
Specify the name of the message retry exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Message retry exit data (MSGRTYDATA)

Specifies user data that is passed to the message retry exit program.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data for the message retry exit program is not specified.

message-retry-exit-user-data
Specify a maximum of 32 characters of user data that is passed to the message retry exit program.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Number of message retries (MSGRTYNBR)

Specifies the number of times the channel will retry before it decides it cannot deliver the message.

This parameter is used by the channel as an alternative to a message retry exit when MSGRTYEXIT is defined as *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

message-retry-number
Specify a value ranging from 0 through 999999999. A value of 0 indicates no retries will be performed.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Message retry interval (MSGRTYITV)

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

This parameter is used by the channel as an alternative to a message retry exit when MSGRTYEXIT is defined as *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

message-retry-number
Specify a value ranging from 0 through 999999999. A value of 0 indicates that the retry will be performed as soon as possible.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Convert message (CVTMSG)

Specifies whether the application data in the message should be converted before the message is transmitted.

The possible values are:

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

***YES** The application data in the message is converted before sending.

***NO** The application data in the message is not converted before sending.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR, *CLTCN or *SVRCN.

Put authority (PUTAUT)

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

The possible values are:

***SAME**
The attribute is unchanged.

***DFT** No authority check is made before the message is put on the destination queue.

***CTX** The user identifier in the message context information is used to establish authority to put the message.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Sequence number wrap (SEQNUMWRAP)

Specifies the maximum message sequence number. When the maximum is reached, sequence numbers wrap to start again at 1.

Note: The maximum message sequence number is not negotiable; the local and remote channels must wrap at the same number.

The possible values are:

***SAME**
The attribute is unchanged.

sequence-number-wrap-value
Specify a value ranging from 100 through 999999999.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Maximum message length (MAXMSGLEN)

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

The possible values are:

***SAME**
The attribute is unchanged.

maximum-message-length
Specify a value ranging from 0 through 104857600. A value of 0 indicates that the maximum length is unlimited.

Heartbeat interval (HRTBTINTVL)

Specifies the time, in seconds, between heartbeat flows passed from the sending MCA when there are no messages on the transmission queue. The heartbeat exchange gives the receiving MCA the opportunity to quiesce the channel. This applies only to sender, server, cluster sender and cluster receiver (*SDR, *SVR, *CLUSSDR and *CLUSRCVR) channels.

The possible values are:

***SAME**
The attribute is unchanged.

heart-beat-interval
Specify a value ranging from 0 through 999999999. A value of 0 means that no heartbeat exchanges are to take place.

Non Persistent Message Speed (NPMSPEED)

Specifies whether the channel supports fast non persistant messages.

The possible values are:

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

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

***NORMAL**
The channel does not support fast non persistant messages.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Cluster Name (CLUSTER)

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

This parameter is valid only for *CLUSSDR and *CLUSRCVR channels. If the CLUSNL parameter is non-blank, this parameter must be blank.

The possible values are:

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

***NONE**
No cluster name is specififed.

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

Cluster Name List (CLUSNL)

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

This parameter is valid only for *CLUSSDR and *CLUSRCVR channels. If the CLUSTER parameter is non-blank, this parameter must be blank.

The possible values are:

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

***NONE**
No cluster namelist is specififed.

cluster-name-list
The name of the namelist specifying a list of clusters to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

Network Connection Priority (NETPRTY)

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

This parameter is valid only for *CLUSRCVR channels.

The possible values are:

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

network-connection-priority
Specify a value ranging from 0 through 9 where 0 is the lowest priority.

SSL CipherSpec (SSLCIPH)

SSLCIPH specifies the CipherSpec used in SSL channel negotiation. The possible values are:

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

cipherspec
The name of the CipherSpec.

SSL Client Authentication (SSLCAUTH)

SSLCAUTH specifies whether the channel carries out client authentication over SSL. The parameter is used only for channels with SSLCIPH specified.

The possible values are:

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

***REQUIRED**
Client authentication is required.

***OPTIONAL**
Client authentication is optional.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *CLTCN or *CLUSSDR.

SSL Peer name (SSLPEER)

SSLPEER specifies the X500 peer name used in SSL channel negotiation. The possible values are:

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

x500peername
The X500 peer name to use.

Local communication address (LOCLADDR)

Specifies the local communication address for the channel.

This parameter is only valid for *SDR, *SVR, *RQSTR, *CLUSSDR, *CLUSRCVR and *CLTCN channels.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The connection is blank.

local-address
Only valid for transport type TCP/IP. Specify the optional IP address and optional port or port range used for outbound TCP/IP communications. The format is LOCLADDR([ip-addr][(low-port[,high-port])]).

Batch Heartbeat Interval (BATCHHB)

The time in milliseconds used to determine whether batch heartbeating occurs on this channel. Batch heartbeating allows channels to determine whether the remote channel instance is still active before going indoubt. A batch heartbeat will occur if a channel MCA has not communicated with the remote channel within the specified time.

The possible values are:

***SAME**
The attribute is unchanged.

batch-heartbeat-interval
Specify a value ranging from 0 through 999999999. A value of 0 indicates that batch heartbeating is not to be used.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR, *CLTCN or *SVRCN.

Task user identifier (USERID)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

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

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

The possible values are:

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

***NONE**
No user identifier is specified.

user-identifier
Specify the task user identifier.

Password (PASSWORD)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

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

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

The possible values are:

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

***NONE**
No password is specified.

password
Specify the password.

Keep Alive Interval (KAINT)

Specifies the keep alive timing interval for this channel.

The possible values are:

***SAME**
The attribute is unchanged.

***AUTO**
The keep alive interval is calculated based upon the negotiated heartbeat value as follows:

- If the negotiated HBINT is greater than 0, keep alive interval is set to that value plus 60 seconds.
- If the negotiated HBINT is 0, the value used is that specified by the KEEPALIVEOPTIONS statement in the TCP profile configuration data set.

keep-alive-interval
Specify a value ranging from 0 through 99999.

Header Compression (COMPHDR)

The list of header data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (*SDR, *SVR, *CLUSSDR, *CLUSRCVR and *CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No header data compression is performed.

***SYSTEM**
Header data compression is performed.

Message Compression (COMPMSG)

The list of message data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (*SDR, *SVR, *CLUSSDR, *CLUSRCVR and *CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No message data compression is performed.

***RLE** Message data compression is performed using run-length encoding.

***ZLIBFAST**
Message data compression is performed using the zlib compression technique. A fast compression time is preferred.

***ZLIBHIGH**
Message data compression is performed using the zlib compression technique. A high level of compression is preferred.

***ANY** Any compression technique supported by the queue manager can be used. This option is only valid for channel types receiver, requester and server connection (*RCVR, *RQSTR and *SVRCN).

Channel Monitoring (MONCHL)

Controls the collection of online monitoring data.

Online monitoring data is not collected when the queue manager attribute MONCHL is set to *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
The collection of online monitoring data is inherited from the setting of the queue manager attribute MONCHL.

***OFF** Online Monitoring Data collection for this channel is switched off.

***LOW** Monitoring data collection is turned on with a low ratio of data collection.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for a channel type (CHLTYPE) of *CLTCN.

Channel Statistics (STATCHL)

Controls the collection of statistics data.

Statistics data is not collected when the queue manager attribute STATCHL is set to *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
Statistics data collection is based upon the setting of the queue manager attribute STATCHL.

***OFF** Statistics data collection for this channel is switched off.

***LOW** Statistics data collection is turned on with a low ratio of data collection.

***MEDIUM**
Statistics data collection is turned on with a moderate ratio of data collection.

***HIGH**
Statistics data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Cluster Workload Rank (CLWLRANK)

Specifies the cluster workload rank of the channel.

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-rank
The cluster workload rank of the channel in the range 0 through 9.

Cluster Workload Priority (CLWLPRTY)

Specifies the cluster workload priority of the channel.

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-priority
The cluster workload priority of the channel in the range 0 through 9.

Cluster Channel Weight (CLWLWGHT)

Specifies the cluster workload weight of the channel.

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-weight
The cluster workload weight of the channel in the range 1 through 99.

Sharing Conversations (SHARECNV)

Specifies the maximum the number of conversations which can be shared over a particular TCP/IP client channel instance (socket).

This parameter is valid for channels with CHLTYPE defined as *CLTCN or *SVRCN.

The possible values are:

***SAME**
The attribute is unchanged.

0 Specifies no sharing of conversations over a TCP/IP socket. The channel instance runs in a mode prior to that of WebSphere MQ Version 7.0, with regard to:
- Administrator stop-quiesce
- Heartbeating
- Read ahead

1 Specifies no sharing of conversations over a TCP/IP socket. Client heartbeating and read ahead are available, whether in an MQGET call or not, and channel quiescing is more controllable.

shared-conversations
The number of shared conversations in the range 2 through 999999999.

This parameter is only valid for client-connection and server-connection channels.

Note: If the client-connection SHARECNV value does not match the server-connection SHARECNV value, the lower of the two values is used.

Property Control (PROPCTL)

Specifies what happens to properties of messages when the message is about to be sent to a V6 or prior queue manager (a queue manager that does not understand the concept of a property descriptor).

The possible values are:

***SAME**
The attribute is unchanged.

***COMPAT**
If the message contains a property with a prefix of ″mcd.″, ″jms.″, ″usr.″ or ″mqext.″ then all optional message properties, except those in the message descriptor (or extension) will be placed in one or more MQRFH2 headers in the message data before the message is sent to the remote queue manager.

***NONE**
All properties of the message, except those in the message descriptor (or extension), will be removed from the message before the message is sent to the remote queue manager.

***ALL** All properties of the message will be included with the message when it is sent to the remote queue manager. The properties, except those in the message descriptor (or extension), will be placed in one or more MQRFH2 headers in the message data.

Maximum Instances (MAXINST)

Specifies the maximum number of simultaneous instances of an individual server-connection channel.

The possible values are:

***SAME**
The attribute is unchanged.

maximum-instances
The maximum number of simultaneous instances of the channel in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

Maximum Instances Per Client (MAXINSTC)

Specifies the maximum number of simultaneous instances of an individual server-connection channel which can be started from a single client.

The possible values are:

***SAME**
The attribute is unchanged.

maximum-instances-per-client
The maximum number of simultaneous instances of the channel which can be in the started from a single client in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running from individual clients, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

Client Channel Weight (CLNTWGHT)

The client channel weighting attribute is used so client channel definitions can be selected at random based on their weighting when more than one suitable definition is available.

The possible values are:

***SAME**
The attribute is unchanged.

client-channel-weight
The client channel weight in the range 0 through 99.

Client Channel Affinity (AFFINITY)

The channel affinity attribute is used so client applications that connect multiple times using the same queue manager name can choose whether to use the same client channel definition for each connection.

The possible values are:

***SAME**
The attribute is unchanged.

***PREFERRED**
The first connection in a process reading a CCDT creates a list of applicable definitions based on the weighting with any applicable CLNTWGHT(0) definitions first and in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful non CLNTWGHT(0) definitions are moved to the end of the list. CLNTWGHT(0) definitions remain at the start of the list and are selected first for each connection.

***NONE**
The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the weighting with any applicable CLNTWGHT(0) definitions selected first in alphabetical order.

Examples

None

Error messages

Unknown

Change MQ Listener (CHGMQMLSR)

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

The Change MQ Listener (CHGMQMLSR) command changes the specified attributes of an existing MQ listener definition.

Parameters

Keyword	Description	Choices	Notes
LSRNAME	Listener name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3
CONTROL	Listener control	***SAME**, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 4
PORT	Port number	0-65535, ***SAME**	Optional, Positional 5
IPADDR	IP Address	*Character value*, *BLANK, ***SAME**	Optional, Positional 6
BACKLOG	Listener backlog	0-999999999, ***SAME**	Optional, Positional 7

Listener name (LSRNAME)

The name of the listener definition to be changed.

The possible values are:

listener-name
Specify the name of the listener definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Text 'description' (TEXT)

Specifies text that briefly describes the listener definition.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Listener control (CONTROL)

Whether the listener starts automatically when the queue manager is started.

The possible values are:

***SAME**
The attribute is unchanged.

***MANUAL**
The listener is not automatically started or stopped.

***QMGR**
The listener is started and stopped as the queue manager is started and stopped.

***STARTONLY**
The listener is started as the queue manager is started, but is not automatically stopped when the queue manager is stopped.

Port number (PORT)

The port number to be used by the listener.

The possible values are:

***SAME**
The attribute is unchanged.

port-number
The port number to be used.

IP Address (IPADDR)

The IP address to be used by the listener.

The possible values are:

***SAME**
The attribute is unchanged.

ip-addr
The IP address to be used.

Listener backlog (BACKLOG)

The number of concurrent connection requests the listener supports.

The possible values are:

***SAME**
The attribute is unchanged.

backlog
The number of concurrent connection requests supported.

Examples

None

Error messages

Unknown

Change MQ Namelist (CHGMQMNL)

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

The Change MQ Namelist (CHGMQMNL) command changes a list of names in the namelist specified on the selected local queue manager.

Parameters

Keyword	Description	Choices	Notes
NAMELIST	Namelist	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3
NAMES	List of Names	Values (up to 256 repetitions): *Character value*, *BLANKS, ***SAME**, *NONE	Optional, Positional 4

Namelist (NAMELIST)

The name of the namelist to be changed.

namelist
Specify the name of the namelist. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used.

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

Text 'description' (TEXT)

Specifies text that briefly describes the namelist.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

***SAME**
The attribute is unchanged.

description
Specify no more than 64 characters enclosed in apostrophes.

List of Names (NAMES)

List of names. This is the list of names to be created. The names can be of any type, but must conform to the rules for naming MQ objects.

***SAME**
The attribute is unchanged.

namelist
The list to create. An empty list is valid.

Examples

None

Error messages

Unknown

Change MQ Process (CHGMQMPRC)

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

The Change MQ Process (CHGMQMPRC) command changes the specified attributes of an existing MQ process definition.

Parameters

Keyword	Description	Choices	Notes
PRCNAME	Process name	*Character value*	Required, Key, Positional 1

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3
APPTYPE	Application type	*Integer*, ***SAME** , *CICS, *MVS, *IMS, *OS2, *DOS, *UNIX, *QMGR, *OS400, *WINDOWS, *CICS_VSE, *WINDOWS_NT, *VMS, *NSK, *VOS, *IMS_BRIDGE, *XCF, *CICS_BRIDGE, *NOTES_AGENT, *BROKER, *JAVA, *DQM	Optional, Positional 4
APPID	Application identifier	*Character value*, ***SAME**	Optional, Positional 5
USRDATA	User data	*Character value*, ***SAME** , *NONE	Optional, Positional 6
ENVDATA	Environment data	*Character value*, ***SAME** , *NONE	Optional, Positional 7

Process name (PRCNAME)

The name of the process definition to be changed.

The possible values are:

process-name
Specify the name of the process definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Text 'description' (TEXT)

Specifies text that briefly describes the process definition.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Application type (APPTYPE)

The type of application started.

The possible values are:

***SAME**
The attribute is unchanged.

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

***MVS** Represents an MVS™ application.

***IMS** Represents an IMS™ application.

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

***DOS** Represents a DOS application.

***UNIX**
Represents a UNIX application.

***QMGR**
Represents a queue manager.

***OS400**
Represents an i5/OS application.

***WINDOWS**
Represents a Windows application.

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

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

***VMS** Represents a VMS application.

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

***VOS** Represents a VOS application.

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

***XCF** Represents an XCF application.

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

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

***BROKER**
Represents a broker application.

***JAVA** Represents a Java application.

***DQM**
Represents a DQM application.

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

Application identifier (APPID)

Application identifier. This is the name of the application to be started, on the platform for which the command is processing. It is typically a program name and library name.

The possible values are:

***SAME**
The attribute is unchanged.

application-id
The maximum length is 256 characters.

User data (USRDATA)

A character string that contains user information pertaining to the application, as defined by APPID, to start.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data is blank.

user-data
Specify up to 128 characters of user data.

Environment data (ENVDATA)

A character string that contains environment information pertaining to the application, as defined by APPID, to start.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The environment data is blank.

environment-data
The maximum length is 128 characters.

Examples

None

Error messages

Unknown

Change MQ Queue (CHGMQMQ)

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

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

Parameters

Keyword	Description	Choices	Notes
QNAME	Queue name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
QTYPE	Queue type	*Character value*	Optional, Positional 3
FORCE	Force	***NO** , *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
PUTENBL	Put enabled	***SAME** , *NO, *YES	Optional, Positional 6
DFTPTY	Default message priority	0-9, ***SAME**	Optional, Positional 7
DFTMSGPST	Default message persistence	***SAME** , *NO, *YES	Optional, Positional 8

Keyword	Description	Choices	Notes
PRCNAME	Process name	*Character value*, *NONE, ***SAME**	Optional, Positional 9
TRGENBL	Triggering enabled	***SAME** , *NO, *YES	Optional, Positional 10
GETENBL	Get enabled	***SAME** , *NO, *YES	Optional, Positional 11
SHARE	Sharing enabled	***SAME** , *NO, *YES	Optional, Positional 12
DFTSHARE	Default share option	***SAME** , *NO, *YES	Optional, Positional 13
MSGDLYSEQ	Message delivery sequence	***SAME** , *PTY, *FIFO	Optional, Positional 14
HDNBKTCNT	Harden backout count	***SAME** , *NO, *YES	Optional, Positional 15
TRGTYPE	Trigger type	***SAME** , *FIRST, *ALL, *DEPTH, *NONE	Optional, Positional 16
TRGDEPTH	Trigger depth	1-999999999, ***SAME**	Optional, Positional 17
TRGMSGPTY	Trigger message priority	0-9, ***SAME**	Optional, Positional 18
TRGDATA	Trigger data	*Character value*, *NONE, ***SAME**	Optional, Positional 19
RTNITV	Retention interval	0-999999999, ***SAME**	Optional, Positional 20
MAXDEPTH	Maximum queue depth	0-999999999, ***SAME**	Optional, Positional 21
MAXMSGLEN	Maximum message length	0-104857600, ***SAME**	Optional, Positional 22
BKTTHLD	Backout threshold	0-999999999, ***SAME**	Optional, Positional 23
BKTQNAME	Backout requeue name	*Character value*, *NONE, ***SAME**	Optional, Positional 24
INITQNAME	Initiation queue	*Character value*, *NONE, ***SAME**	Optional, Positional 25
USAGE	Usage	***SAME** , *NORMAL, *TMQ	Optional, Positional 26
DFNTYPE	Definition type	***SAME** , *TEMPDYN, *PERMDYN	Optional, Positional 27
TGTQNAME	Target queue	*Character value*, ***SAME**	Optional, Positional 28
RMTQNAME	Remote queue	*Character value*, ***SAME** , *NONE	Optional, Positional 29
RMTMQMNAME	Remote Message Queue Manager	*Character value*, ***SAME**	Optional, Positional 30
TMQNAME	Transmission queue	*Character value*, *NONE, ***SAME**	Optional, Positional 31

Keyword	Description	Choices	Notes
HIGHTHLD	Queue depth high threshold	0-100, ***SAME**	Optional, Positional 32
LOWTHLD	Queue depth low threshold	0-100, ***SAME**	Optional, Positional 33
FULLEVT	Queue full events enabled	***SAME** , *NO, *YES	Optional, Positional 34
HIGHEVT	Queue high events enabled	***SAME** , *NO, *YES	Optional, Positional 35
LOWEVT	Queue low events enabled	***SAME** , *NO, *YES	Optional, Positional 36
SRVITV	Service interval	0-999999999, ***SAME**	Optional, Positional 37
SRVEVT	Service interval events	***SAME** , *HIGH, *OK, *NONE	Optional, Positional 38
DISTLIST	Distribution list support	***SAME** , *NO, *YES	Optional, Positional 39
CLUSTER	Cluster Name	*Character value*, ***SAME** , *NONE	Optional, Positional 40
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SAME**	Optional, Positional 41
DEFBIND	Default Binding	***SAME** , *OPEN, *NOTFIXED	Optional, Positional 42
CLWLRANK	Cluster Workload Rank	0-9, ***SAME**	Optional, Positional 43
CLWLPRTY	Cluster Workload Priority	0-9, ***SAME**	Optional, Positional 44
CLWLUSEQ	Cluster workload queue use	***SAME** , *QMGR, *LOCAL, *ANY	Optional, Positional 45
MONQ	Queue Monitoring	***SAME** , *QMGR, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 46
STATQ	Queue Statistics	***SAME** , *QMGR, *OFF, *ON	Optional, Positional 47
ACCTQ	Queue Accounting	***SAME** , *QMGR, *OFF, *ON	Optional, Positional 48
NPMCLASS	Non Persistent Message Class	***SAME** , *NORMAL, *HIGH	Optional, Positional 49
MSGREADAHD	Message Read Ahead	***SAME** , *DISABLED, *NO, *YES	Optional, Positional 50
DFTPUTRESP	Default Put Response	***SAME** , *SYNC, *ASYNC	Optional, Positional 51
PROPCTL	Property Control	***SAME** , *COMPAT, *NONE, *ALL, *FORCE	Optional, Positional 52
TARGTYPE	Target Type	***SAME** , *QUEUE, *TOPIC	Optional, Positional 53

Queue name (QNAME)

The name of the queue to be changed.

The possible values are:

queue-name
Specify the name of the queue.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Queue type (QTYPE)

Specifies the type of queue that is to be changed.

The possible values are:

***ALS** An alias queue.

***LCL** A local queue.

***RMT** A remote queue.

***MDL** A model queue.

Force (FORCE)

Specifies whether the command should be forced to complete when conditions are such that completing the command affects an open queue. The conditions depend on the type of the queue that is being changed:

Alias Queue
The TGTQNAME keyword is specified with a queue name and an application has the alias queue open.

Local Queue
Either of the following conditions indicate that a local queue will be affected:

- SHARE(*NO) is specified and more than one application has the local queue open for input.

- The USAGE attribute is changed and one or more applications has the local queue open, or, there are one or more messages on the queue. (The USAGE attribute should not normally be changed while there are messages on the queue; the format of messages changes when they are put on a transmission queue.)

Remote Queue
Either of the following conditions indicate that a remote queue will be affected:

- The TMQNAME keyword is specified with a transmission-queue name (or *NONE) and an application with the remote queue open will be affected by this change.
- Any of the RMTQNAME, RMTMQMNAME or TMQNAME keywords is specified with a queue or queue manager name, and one or more applications has a queue open that resolves through this definition as a queue manager alias.

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

The possible values are:

***NO** The command fails if the relevant conditions are true.

***YES** The command is forced to complete successfully even if the relevant conditions are true.

Text 'description' (TEXT)

Specifies text that briefly describes the queue definition.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Put enabled (PUTENBL)

Specifies whether messages can be put on the queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Messages cannot be added to the queue.

***YES** Messages can be added to the queue by authorized applications.

Default message priority (DFTPTY)

Specifies the default priority of messages put on the queue.

The possible values are:

***SAME**
The attribute is unchanged.

priority-value
Specify a value ranging from 0 through 9, where 9 is the highest priority.

Default message persistence (DFTMSGPST)

Specifies the default for message-persistence on the queue. Message persistence determines whether or not messages are preserved across restarts of the queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** By default, messages are lost across a restart of the queue manager.

***YES** By default, messages are preserved across a restart of the queue manager.

Process name (PRCNAME)

Specifies the local name of the MQ process that identifies the application that should be started when a trigger event occurs.

The process does not have to be available when the queue is created, but it must be available for a trigger event to occur.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The process name is blank.

process-name
Specify the name of the MQ process.

Triggering enabled (TRGENBL)

Specifies whether trigger messages are written to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Triggering is not enabled. Trigger messages are not written to the initiation queue.

***YES** Triggering is enabled. Trigger messages are written to the initiation queue.

Get enabled (GETENBL)

Specifies whether applications are to be permitted to get messages from this queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Applications cannot retrieve messages from the queue.

***YES** Suitably authorized applications can retrieve messages from the queue.

Sharing enabled (SHARE)

Specifies whether multiple instances of applications can open this queue for input simultaneously.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Only a single application instance can open the queue for input.

***YES** More than one application instance can open the queue for input.

Default share option (DFTSHARE)

Specifies the default share option for applications opening this queue for input.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** By default, the open request is for exclusive use of the queue for input.

***YES** By default, the open request is for shared use of the queue for input.

Message delivery sequence (MSGDLYSEQ)

Specifies the message delivery sequence.

The possible values are:

***SAME**
The attribute is unchanged.

***PTY** Messages are delivered in first-in-first-out (FIFO) order within priority.

***FIFO** Messages are delivered in FIFO order regardless of priority.

Harden backout count (HDNBKTCNT)

Specifies whether the count of backed out messages is saved (hardened) across restarts of the message queue manager.

Note: On WebSphere MQ for i5/OS the count is ALWAYS hardened, regardless of the setting of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** The backout count is not hardened.

***YES** The backout count is hardened.

Trigger type (TRGTYPE)

Specifies the condition that initiates a trigger event. When the condition is true, a trigger message is sent to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***FIRST**
When the number of messages on the queue goes from 0 to 1.

***ALL** Every time a message arrives on the queue.

***DEPTH**
When the number of messages on the queue equals the value of the TRGDEPTH attribute.

***NONE**
No trigger messages are written.

Trigger depth (TRGDEPTH)

Specifies, for TRIGTYPE(*DEPTH), the number of messages that initiate a trigger message to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

depth-value
Specify a value ranging from 1 through 999999999.

Trigger message priority (TRGMSGPTY)

Specifies the minimum priority that a message must have before it can produce, or be counted for, a trigger event.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

priority-value
Specify a value ranging from 0 through 9, where 9 is the highest priority.

Trigger data (TRGDATA)

Specifies up to 64 characters of user data that the queue manager includes in the trigger message. This data is made available to the monitoring application that processes the initiation queue, and to the application started by the monitor.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No trigger data is specified.

trigger-data
Specify up to 64 characters enclosed in apostrophes. For a transmission queue you can use this parameter to specify the name of the channel to be started.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Retention interval (RTNITV)

Specifies the retention interval. This interval is the number of hours for which the queue might be needed, based on the date and time when the queue was created.

This information is available to a housekeeping application or an operator and can be used to determine when a queue is no longer required.

Note: The message queue manager does not delete queues, nor does it prevent your queues from being deleted if their retention interval has not expired. It is your responsibility to take any required action.

The possible values are:

***SAME**
The attribute is unchanged.

interval-value
Specify a value ranging from 0 through 999999999.

Maximum queue depth (MAXDEPTH)

Specifies the maximum number of messages allowed on the queue. However, other factors can cause the queue to be treated as full; for example, it appears to be full if there is no storage available for a message.

Note: If this value is subsequently reduced by using the CHGMQMQ command, any messages that are on the queue remain intact even if they cause the new maximum to be exceeded.

The possible values are:

***SAME**
The attribute is unchanged.

depth-value
Specify a value ranging from 0 through 999999999.

Maximum message length (MAXMSGLEN)

Specifies the maximum length for messages on the queue.

Note: If this value is subsequently reduced by using the CHGMQMQ command, any messages that are on the queue remain intact even if they exceed the new maximum length.

Applications might use the value of this attribute to determine the size of buffer they need to retrieve messages from the queue. Therefore change the value only if you know this will not cause an application to operate incorrectly.

The possible values are:

***SAME**
The attribute is unchanged.

length-value
Specify a value ranging from 0 through 100 MB in bytes. The default is 4MB.

Backout threshold (BKTTHLD)

Specifies the backout threshold. WebSphere MQ for i5/OS takes no action based on the value of this attribute, excpet allowing for this attribute to be queried.

The possible values are:

***SAME**
The attribute is unchanged.

threshold-value
Specify a value ranging from 0 through 999999999.

Backout requeue name (BKTQNAME)

Specifies the backout-queue name. WebSphere MQ for i5/OS takes no action based on the value of this attribute, except allowing for this attribute to be queried.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No backout queue is specified.

backout-queue-name
Specify the backout queue name.

Initiation queue (INITQNAME)

Specifies the name of the initiation queue.

Note: The initiation queue must be on the same instance of a message queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No initiation queue is specified.

initiation-queue-name
Specify the initiation queue name.

Usage (USAGE)

Specifies whether the queue is for normal usage, or for transmitting messages to a remote message queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

***NORMAL**
Normal usage (the queue is not a transmission queue)

***TMQ** The queue is a transmission queue that is used to hold messages destined for a remote message queue manager. If the queue is intended for use in situations where a transmission queue name is not explicitly specified, the queue name must be the same as the name of the remote message queue manager. For further information, see WebSphere MQ Intercommunication.

Definition type (DFNTYPE)

Specifies the type of dynamic queue definition that is created when an application issues an MQOPEN API call with the name of this model queue specified in the object descriptor.

Note: This parameter only applies to a model queue definition.

The possible values are:

***SAME**
The attribute is unchanged.

***TEMPDYN**
A temporary dynamic queue is created. This value should not be specified with a DEFMSGPST value of *YES.

***PERMDYN**
A permanent dynamic queue is created.

Target queue (TGTQNAME)

Specifies the name of the object for which this queue is an alias.

The object can be a local or remote queue, a topic or a message queue manager.

Note: The target object does not need to exist at this time but it must exist when a process attempts to open the alias queue.

The possible values are:

***SAME**
The attribute is unchanged.

target-object-name
Specify the name of the target object.

Remote queue (RMTQNAME)

Specifies the name of the remote queue. That is, the local name of the remote queue as defined on the queue manager specified by RMTMQMNAME.

If this definition is used for a queue manager alias definition, RMTQNAME must be blank when the open occurs.

If this definition is used for a reply-to alias, this name is the name of the queue that is to be the reply-to queue.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No remote-queue name is specified (that is, the name is blank). This can be used if the definition is a queue manager alias definition.

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

Note: The name is not checked to ensure that it contains only those characters normally allowed for queue names.

Remote Message Queue Manager (RMTMQMNAME)

Specifies the name of the remote queue manager on which the queue RMTQNAME is defined.

If an application opens the local definition of a remote queue, RMTMQMNAME must not be the name of the connected queue manager. If TMQNAME is blank there must be a local queue of this name, which is to be used as the transmission queue.

If this definition is used for a queue manager alias, RMTMQMNAME is the name of the queue manager, which can be the name of the connected queue manager. Otherwise, if TMQNAME is blank, when the queue is opened there must be a local queue of this name, with USAGE(*TMQ) specified, which is to be used as the transmission queue.

If this definition is used for a reply-to alias, this name is the name of the queue manager that is to be the reply-to queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

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

Note: Ensure this name contains only those characters normally allowed for queue manager names.

Transmission queue (TMQNAME)

Specifies the local name of the transmission queue to be used for messages destined for the remote queue, for either a remote queue or for a queue manager alias definition.

If TMQNAME is blank, a queue with the same name as RMTMQMNAME is used as the transmission queue.

This attribute is ignored if the definition is being used as a queue manager alias and RMTMQMNAME is the name of the connected queue manager.

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

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No specific transmission queue name is defined for this remote queue. The value of this attribute is set to all blanks.

transmission-queue-name
Specify the transmission queue name.

Queue depth high threshold (HIGHTHLD)

Specifies the threshold against which the queue depth is compared to generate a queue depth high event.

The possible values are:

***SAME**
The attribute is unchanged.

threshold-value
Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

Queue depth low threshold (LOWTHLD)

Specifies the threshold against which the queue depth is compared to generate a queue depth low event.

The possible values are:

***SAME**
The attribute is unchanged.

threshold-value
Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

Queue full events enabled (FULLEVT)

Specifies whether queue full events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Queue full events are not generated.

***YES** Queue full events are generated.

Queue high events enabled (HIGHEVT)

Specifies whether queue depth high events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Queue depth high events are not generated.

***YES** Queue depth high events are generated.

Queue low events enabled (LOWEVT)

Specifies whether queue depth low events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Queue depth low events are not generated.

***YES** Queue depth low events are generated.

Service interval (SRVITV)

Specifies the service interval. This interval is used for comparison to generate service interval high and service interval OK events.

The possible values are:

***SAME**
The attribute is unchanged.

interval-value
Specify a value ranging from 0 through 999999999. The value is in units of milliseconds.

Service interval events (SRVEVT)

Specifies whether service interval high or service interval OK events are generated.

A service interval high event is generated when a check indicates that no messages have been retrieved from the queue for the time indicated by the SRVITV parameter as a minimum.

A service interval OK event is generated when a check indicates that messages have been retrieved from the queue within the time indicated by the SRVITV parameter.

The possible values are:

***SAME**
The attribute is unchanged.

***HIGH**
Service interval high events are generated.

***OK** Service interval OK events are generated.

***NONE**
No service interval events are generated.

Distribution list support (DISTLIST)

Specifies whether the queue supports distribution lists.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** The queue will not support distribution lists.

***YES** The queue will support distribution lists.

Cluster Name (CLUSTER)

The name of the cluster to which the queue belongs.

Changes to this parameter do not affect instances of the queue that are already open.

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

The possible values are:

***SAME**
The attribute is unchanged.

cluster-name
Only one of the resultant values of CLUSTER or CLUSNL can be non-blank; you cannot specify a value for both.

Cluster Name List (CLUSNL)

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

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

The possible values are:

***SAME**
The attribute is unchanged.

namelist-name
Only one of the resultant values of CLUSTER or CLUSNL can be non-blank; you cannot specify a value for both.

Default Binding (DEFBIND)

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

The possible values are:

***SAME**
The attribute is unchanged.

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

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

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

Cluster Workload Rank (CLWLRANK)

Specifies the cluster workload rank of the queue.

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-rank
Specify a value ranging from 0 through 9.

Cluster Workload Priority (CLWLPRTY)

Specifies the cluster workload priority of the queue.

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-priority
Specify a value ranging from 0 through 9.

Cluster workload queue use (CLWLUSEQ)

Specifies the behaviour of an MQPUT when the target queue has both a local instance and at least one remote cluster instance. If the put originates from a cluster channel then this attribute does not apply.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
The value is inherited from the Queue Manager CLWLUSEQ attribute.

***LOCAL**
The local queue will be the sole target of the MQPUT.

***ANY** The queue manager will treat such a local queue as another instance of the cluster queue for the purposes of workload distribution.

Queue Monitoring (MONQ)

Controls the collection of Online Monitoring Data.

Online Monitoring Data is not collected when the queue manager attribute MONQ is set to *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
The collection of online monitoring data is inherited from the setting of the queue manager attribute MONQ.

***OFF** Online monitoring data collection for this queue is switched off.

***LOW** Monitoring data collection is turned on with a low ratio of data collection.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection.

Queue Statistics (STATQ)

Controls the collection of statistics data.

Online monitoring data is not collected when the queue manager attribute STATQ is set to *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
Statistics data collection is based upon the setting of the queue manager attribute STATQ.

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

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

Queue Accounting (ACCTQ)

Controls the collection of accouting data.

Accounting data is not collected when the queue manager attribute ACCTQ is set to *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

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

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

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

Non Persistent Message Class (NPMCLASS)

Specifies the level of reliability for non-persistent messages put to this queue.

The possible values are:

***SAME**
The attribute is unchanged.

***NORMAL**
Non-persistent messages put to this queue are only lost following a failure, or a queue manager shutdown. Non-persistent message put to this queue will be discarded in the event of a queue manager restart.

***HIGH**
Non-persistent messages put to this queue are not discarded in the event of a queue manager restart. Non-persistent messages put to this queue may still be lost in the event of a failure.

Message Read Ahead (MSGREADAHD)

Specifies whether non persistent messages are sent to the client ahead of an application requesting them.

The possible values are:

***SAME**
The attribute is unchanged.

***DISABLED**
Read ahead is disabled for this queue. Messages are not sent to the client ahead of an application requesting them regardless of whether read ahead is requested by the client application.

***NO** Non-persistent messages are not sent to the client ahead of an application requesting them. A maximum of one non-persistent message can be lost if the client ends abnormally.

***YES** Non-persistent messages are sent to the client ahead of an application requesting them. Non-persistent messages can be lost if the client ends abnormally or if the client application does not consume all the messages it is sent.

Default Put Response (DFTPUTRESP)

The default put response type (DFTPUTRESP) attribute specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO_RESPONSE_AS_Q_DEF option.

The possible values are:

***SAME**
The attribute is unchanged.

***SYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are issued as if MQPMO_SYNC_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application. This is the default value supplied with WebSphere MQ, but your installation might have changed it.

***ASYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are always issued as if MQPMO_ASYNC_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application; but an improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

Property Control (PROPCTL)

Specifies what happens to properties of messages that are retrieved from queues using the MQGET call when the MQGMO_PROPERTIES_AS_Q_DEF option is specified.

The possible values are:

***SAME**
The attribute is unchanged.

***COMPAT**
If the message contains a property with a prefix of ″mcd.″, ″jms.″, ″usr.″ or ″mqext.″ then all message properties are delivered to the application in an MQRFH2 header. Otherwise all properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

***NONE**
All properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

***ALL** All properties of the message, except those contained in the message descriptor (or extension), are contained in one or more MQRFH2 headers in the message data.

***FORCE**
Properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle.

Target Type (TARGTYPE)

Specifies the type of object to which the alias resolves.

The possible values are:

***SAME**
The attribute is unchanged.

***QUEUE**
Queue object.

***TOPIC**
Topic object.

Examples

None

Error messages

Unknown

Change MQ Subscription (CHGMQMSUB)

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

The Change MQ Subscription (CHGMQMSUB) command changes the specified attributes of an existing MQ subscription.

Parameters

Keyword	Description	Choices	Notes
SUBID	Subscription identifier	*Character value*, ***SAME**	Optional, Key, Positional 1
SUBNAME	Subscription name	*Character value*, ***SAME**	Optional, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
TOPICSTR	Topic string	*Character value*, *NONE, ***SAME**	Optional, Positional 4
TOPICOBJ	Topic object	*Character value*, *NONE, ***SAME**	Optional, Positional 5
DEST	Destination	*Character value*, ***SAME**	Optional, Positional 6
DESTMQM	Destination Queue Manager	*Character value*, *NONE, ***SAME**	Optional, Positional 7
DESTCRRLID	Destination Correlation Id	*Character value*, *NONE, ***SAME**	Optional, Positional 8
PUBACCT	Publish Accounting Token	*Character value*, *NONE, ***SAME**	Optional, Positional 9
PUBAPPID	Publish Application Id	*Character value*, *NONE, ***SAME**	Optional, Positional 10
SUBUSER	Subscription User Id	*Character value*, *NONE, ***SAME**	Optional, Positional 11
USERDATA	Subscription User Data	*Character value*, *NONE, ***SAME**	Optional, Positional 12
SELECTOR	Selector String	*Character value*, *NONE, ***SAME**	Optional, Positional 13
PSPROP	PubSub Property	***SAME**, *NONE, *COMPAT, *RFH2	Optional, Positional 14
DESTCLASS	Destination Class	***SAME**, *MANAGED, *PROVIDED	Optional, Positional 15
SUBSCOPE	Subscription Scope	***SAME**, *ALL, *QMGR	Optional, Positional 16
VARUSER	Variable User	***SAME**, *ANY, *FIXED	Optional, Positional 17

Keyword	Description	Choices	Notes
REQONLY	Request Publications	***SAME**, *YES, *NO	Optional, Positional 18
PUBPTY	Publish Priority	0-9, ***SAME**, *ASPUB, *ASQDEF	Optional, Positional 19
WSCHEMA	Wildcard Schema	***SAME**, *CHAR, *TOPIC	Optional, Positional 20
EXPIRY	Expiry Time	0-999999999, ***SAME**, *UNLIMITED	Optional, Positional 21

Subscription identifier (SUBID)

The subscription identifier of the subscription to be changed.

The possible values are:

subscription-identifier
Specify the 48 character hexadecimal string representing the 24 byte subscription identifier.

Subscription name (SUBNAME)

The name of the subscription to be changed.

The possible values are:

subscription-name
Specify a maximum of 256 bytes for the subscription name.

Note: Subscription names of greater than 256 bytes can be speicifed using MQSC.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Topic string (TOPICSTR)

Specifies the topic string associated with this subscription.

The possible values are:

topic-string
Specify a maximum of 256 bytes for the topic string.

Note: Topic strings of greater than 256 bytes can be speicifed using MQSC.

Topic object (TOPICOBJ)

Specifies the topic object associated with this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

topic-object
Specify the name of the topic object.

Destination (DEST)

Specifies the destination queue for messages published to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

destination-queue
Specify the name of the destination queue.

Destination Queue Manager (DESTMQM)

Specifies the destination queue manager for messages published to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No destination queue manager is specified.

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

Destination Correlation Id (DESTCRRLID)

Specifies the correlation identifier for messages published to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Messages are placed on the destination with a correlation identifier of MQCI_NONE.

correlation-identifier
Specify the 48 character hexadecimal string representing the 24 byte correlation identifier.

Publish Accounting Token (PUBACCT)

Specifies the accounting token for messages published to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Messages are placed on the destination with an accounting token of MQACT_NONE.

publish-accounting-token
Specify the 64 character hexadecimal string representing the 32 byte publish accounting token.

Publish Application Id (PUBAPPID)

Specifies the publish application identity for messages published to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No publish application identifier is specified.

publish-application-identifier
Specify the publish application identifier.

Subscription User Id (SUBUSER)

Specifies the user profile that 'owns' this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No user profile is specified.

user-profile
Specify the user profile.

Subscription User Data (USERDATA)

Specifies the user data associated with the subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No user data is specified.

user-data
Specify a maximum of 256 bytes for user data.

Note: User data of greater than 256 bytes can be speicifed using MQSC.

Selector String (SELECTOR)

Specifies the SQL 92 selector string to be applied to messages published on the named topic to select whether they are eligible for this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No selection string is specified.

selection-string
Specify a maximum of 256 bytes for selection string.

Note: Selection strings of greater than 256 bytes can be speicifed using MQSC.

PubSub Property (PSPROP)

Specifies the manner in which publish / subscribe related message properties are added to messages sent to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Publish / subscribe properties are not added to the message.

***COMPAT**
Publish / subscribe properties are added to the message to maintain compatibility with V6 Publish / Subscribe.

***RFH2**
Publish / subscribe properties are added to the message within an RFH Version 2 header.

Destination Class (DESTCLASS)

Specifies whether this is a managed subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***MANAGED**
The destination is managed.

***PROVIDED**
The destination is a queue.

Subscription Scope (SUBSCOPE)

Specifies whether this subscription should be forwarded (as a proxy subscription) to other brokers, so that the subscriber will receive messages published at those other brokers.

The possible values are:

***SAME**
The attribute is unchanged.

***ALL** The subscription will be forwarded to all queue managers directly connected via a publish / subscribe collective or hierarchy.

***QMGR**
The subscription will only forward messages published on the topic within this queue manager.

Variable User (VARUSER)

Specifies whether user profiles other than the creator of the subscription can connect to it (subject to topic and destination authority checks).

The possible values are:

***SAME**
The attribute is unchanged.

***ANY** Any user profiles can connect to the subsciption.

***FIXED**
Only the user profile that created the subscription can connect to it.

Request Publications (REQONLY)

Specifies whether the subscriber will poll for updates via MQSUBRQ API, or whether all publications are delivered to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***YES** Publications are only delivered to this subscription in response to an MQSUBRQ API.

***NO** All publications on the topic are delivered to this subscription.

Publish Priority (PUBPTY)

Specifies the priority of the message sent to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPUB**
The priority of the message sent to this subscription is taken from that supplied in the published message.

***ASQDEF**
The priority of the message sent to this subscription is taken from the default priority of the queue defined as the destination.

priority-value
Specify a priority ranging from 0 through 9.

Wildcard Schema (WSCHEMA)

Specifies the schema to be used when interpreting wild card characters in the topic string.

The possible values are:

***SAME**
The attribute is unchanged.

***TOPIC**
Wildcard characters represent portions of the topic hierarchy.

***CHAR**
Wildcard characters represent portions of strings.

Expiry Time (EXPIRY)

Specifies the expiry time of the subscription. After a subscription's expiry time has elapsed, it becomes eligible to be discarded by the queue manager and will receive no further publications.

The possible values are:

***SAME**
The attribute is unchanged.

***UNLIMITED**
The subscription does not expire.

expiry-time
Specify an expiry time in tenths of a second ranging from 0 through 999999999.

Examples

None

Error messages

Unknown

Change MQ Service (CHGMQMSVC)

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

The Change MQ Service (CHGMQMSVC) command changes the specified attributes of an existing MQ service definition.

Parameters

Keyword	Description	Choices	Notes
SVCNAME	Service name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3
STRCMD	Start program	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 4
	Qualifier 1: Start program	*Name*	
	Qualifier 2: Library	*Name*	
STRARG	Start program arguments	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
ENDCMD	End program	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 6
	Qualifier 1: End program	*Name*	
	Qualifier 2: Library	*Name*	
ENDARG	End program arguments	*Character value*, *BLANK, ***SAME**	Optional, Positional 7
STDOUT	Standard output	*Character value*, *BLANK, ***SAME**	Optional, Positional 8
STDERR	Standard error	*Character value*, *BLANK, ***SAME**	Optional, Positional 9
TYPE	Service type	***SAME**, *CMD, *SVR	Optional, Positional 10
CONTROL	Service control	***SAME**, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 11

Service name (SVCNAME)

The name of the service definition to be changed.

The possible values are:

service-name
Specify the name of the service definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Text 'description' (TEXT)

Specifies text that briefly describes the service definition.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Start program (STRCMD)

The name of the program to run.

The possible values are:

***SAME**
The attribute is unchanged.

start-command
The name of the start command executable.

Start program arguments (STRARG)

The arguments passed to the program at startup.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
No arguments are passed to the start command.

start-command-arguments
The arguments passed to the start command.

End program (ENDCMD)

The name of the executable to run when the service is requested to stop.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
No end command is executed.

end-command
The name of the end command executable.

End program arguments (ENDARG)

The arguments passed to the end program when the service is requested to stop.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
No arguments are passed to the end command.

end-command-arguments
The arguments passed to the end command.

Standard output (STDOUT)

The path to a file to which the standard output of the service program is redirected.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The standard output is discarded.

stdout-path
The standard output path.

Standard error (STDERR)

The path to a file to which the standard error of the service program is redirected.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The standard error is discarded.

stderr-path
The standard error path.

Service type (TYPE)

Mode in which to run service.

The possible values are:

***SAME**
The attribute is unchanged.

***CMD** When started the command is executed but no status is collected or displayed.

***SVR** The status of the executable started will be monitored and displayed.

Service control (CONTROL)

Whether the service should be started automatically at queue manager start.

The possible values are:

***SAME**
The attribute is unchanged.

***MANUAL**
The service is automatically started or stopped.

***QMGR**
The service is started and stopped as the queue manager is started and stopped.

***STARTONLY**
The service is started as the queue manager is started, but will not be requested to stop when the queue manager is stopped.

Examples

None

Error messages

Unknown

Change MQ Topic (CHGMQMTOP)

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

The Change MQ Topic (CHGMQMTOP) command changes the specified attributes of an existing MQ topic object.

Parameters

Keyword	Description	Choices	Notes
TOPNAME	Topic name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 3
TOPICSTR	Topic string	*Character value*, *BLANK, ***SAME**	Optional, Positional 4
DURSUB	Durable subscriptions	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 5
MGDDURMDL	Durable model queue	*Character value*, *NONE, ***SAME**	Optional, Positional 6
MGDNDURMDL	Non-durable model queue	*Character value*, *NONE, ***SAME**	Optional, Positional 7
PUBENBL	Publish	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 8
SUBENBL	Subscribe	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 9
DFTPTY	Default message priority	0-9, ***SAME**, *ASPARENT	Optional, Positional 10
DFTMSGPST	Default message persistence	***SAME**, *ASPARENT, *YES, *NO	Optional, Positional 11
DFTPUTRESP	Default Put Response	***SAME**, *ASPARENT, *SYNC, *ASYNC	Optional, Positional 12
WILDCARD	Wildcard behaviour	***SAME**, *PASSTHRU, *BLOCK	Optional, Positional 13

Keyword	Description	Choices	Notes
PMSGDLV	Persistent message delivery	***SAME**, *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 14
NPMSGDLV	Non-persistent message deliver	***SAME**, *ASPARENT, *ALL, *ALLDUR, *ALLAVAIL	Optional, Positional 15

Topic name (TOPNAME)

The name of the topic object to be changed.

The possible values are:

topic-name
Specify the name of the topic object. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Text 'description' (TEXT)

Specifies text that briefly describes the topic object.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Topic string (TOPICSTR)

Specifies the topic string represented by this topic object definition.

The possible values are:

***SAME**
The attribute is unchanged.

topic-string
Specify a maximum of 256 bytes for the topic string.

Note: Topic strings of greater than 256 bytes can be speicifed using MQSC.

Durable subscriptions (DURSUB)

Specifies whether applications are permitted to make durable subscriptions on this topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
Whether durable subscriptions can be made on this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***YES** Durable subscriptions can be made on this topic.

***NO** Durable subscriptions cannot be made on this topic.

Durable model queue (MGDDURMDL)

Specifies the name of the model queue to be used for durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

***SAME**
The attribute is unchanged.

durable-model-queue
Specify the name of the model queue.

Non-durable model queue (MGDNDURMDL)

Specifies the name of the model queue to be used for non-durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

***SAME**
The attribute is unchanged.

non-durable-model-queue
Specify the name of the model queue.

Publish (PUBENBL)

Specifies whether messages can be published to the topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
Whether messages can be published to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***YES** Messages can be published to the topic.

***NO** Messages cannot be published to the topic.

Subscribe (SUBENBL)

Specifies whether applications are to be permitted to subscribe to this topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
Whether applications can subscribe to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***YES** Subscriptions can be made to this topic.

***NO** Applications cannot subscribe to this topic.

Default message priority (DFTPTY)

Specifies the default priority of messages published to the topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
The default priority is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

priority-value
Specify a value ranging from 0 through 9.

Default message persistence (DFTMSGPST)

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

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
The default persistence is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

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

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

Default Put Response (DFTPUTRESP)

Specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO_RESPONSE_AS_Q_DEF option.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
The default response type is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***SYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are issued as if MQPMO_SYNC_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application.

***ASYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are always issued as if MQPMO_ASYNC_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application. An improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

Wildcard behaviour (WILDCARD)

Specifies the behaviour of wildcard subscriptions with respect to this topic.

The possible values are:

***SAME**
The attribute is unchanged.

***PASSTHRU**
Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will receive publications made to this topic and to topic strings more specific than this topic.

***BLOCK**
Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will not receive publications made to this topic or to topic strings more specific than this topic.

Persistent message delivery (PMSGDLV)

Specifies the delivery mechanism for persistent messages published to this topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***ALL** Persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

***ALLDUR**
Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

***ALLAVAIL**
Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

Non-persistent message delivery (NPMSGDLV)

Specifies the delivery mechanism for non-persistent messages published to this topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***ALL** Non-persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

***ALLDUR**
Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

***ALLAVAIL**
Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

Examples

None

Error messages

Unknown

Clear MQ Pub/Sub Broker (CLRMQMBRK)

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

The Clear WebSphere MQ broker (CLRMQMBRK) is used to clear the brokers memory of a neighbouring broker. The broker cancels all subscriptions from the target broker. The broker must be stopped when this command is issued. The command is synchronous, and when it has completed the broker can be restarted normally. No messages are read from any of the input queues. After restart, the broker detects any messages on its input queues that came from this broker, and processes them according to their report options.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*	Required, Positional 1
BRKPARENT	Break Parent link	***NO** , *YES	Optional, Positional 2
CHILDMQM	Child Message Queue Manager	*Character value*	Optional, Positional 3

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

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

Break Parent link (BRKPARENT)

Specifies how the broker is ended.

The possible values are:

***YES** Specifies that the link is to be broken with the parent broker. If you specify this parameter you must not specify a value for CHILDMQM.

***NO** Specifies that the link is to be broken with a child broker. Use the CHILDMQM parameter to specify the name of the queue manager that hosts the child broker.

Child Message Queue Manager (CHILDMQM)

Specifies the name of the queue manager that hosts the child broker that the link is to be broken with.

Examples

None

Error messages

Unknown

Clear MQ Queue (CLRMQMQ)

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

The Clear MQ Queue (CLRMQMQ) command deletes all of the messages from a local queue.

The command fails if the queue contains uncommitted messages, or if an application has the queue open.

Parameters

Keyword	Description	Choices	Notes
QNAME	Queue name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Queue name (QNAME)

The name of the queue to be cleared.

The possible values are:

queue-name
Specify the name of the queue.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Examples

None

Error messages

Unknown

Clear MQ Topic String (CLRMQMTOP)

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

The Clear MQ Topic String (CLRMQMTOP) command clears the specified topic string.

Parameters

Keyword	Description	Choices	Notes
TOPICSTR	Topic string	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
CLRTYPE	Clear type	***RETAINED**	Optional, Positional 3
SCOPE	Clear scope	***LOCAL**, *GLOBAL	Optional, Positional 4

Topic string (TOPICSTR)

The topic string to be cleared.

The possible values are:

topic-string
Specify a maximum of 256 bytes for the topic string.

Note: Topic strings of greater than 256 bytes can be speicifed using MQSC.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Clear type (CLRTYPE)

The type of clear topic string to be performed.

The value must be:

***RETAINED**
Remove the retained publication from the specified topic string.

Clear scope (SCOPE)

The scope of the deletion of retained messages.

The possible values are:

***LOCAL**
The retained message is removed from the specified topic string at the local queue manager only.

***GLOBAL**
The retained message is removed from the specified topic string at all queue managers connected in the pub/sub cluster.

Examples

None

Error messages

Unknown

Copy MQ AuthInfo object (CPYMQMAUTI)

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

The Copy MQ AuthInfo object (CPYMQMAUTI) command creates an authentication information object of the same type and, for attributes not specified in the command, with the same attribute values as an existing object.

Parameters

Keyword	Description	Choices	Notes
FROMAI	From AuthInfo name	*Character value*	Required, Key, Positional 1
TOAI	To AuthInfo name	*Character value*	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
AUTHTYPE	AuthInfo type	*CRLLDAP	Optional, Positional 4
CONNAME	Connection name	*Character value*, ***SAME**	Optional, Positional 5
REPLACE	Replace	***NO**, *YES	Optional, Positional 6
TEXT	Text 'description'	*Character value*, ***SAME**, *NONE	Optional, Positional 7
USERNAME	User name	*Character value*, ***SAME**, *NONE	Optional, Positional 8
PASSWORD	User password	*Character value*, ***SAME**, *NONE	Optional, Positional 9

From AuthInfo name (FROMAI)

The name of an existing authentication information object to provide values for the attributes not specified in this command.

The possible values are:

authentication-information-name
Specify the name of the authentication information object. The maximum string length is 48 characters.

To AuthInfo name (TOAI)

The name of the new authentication information object to create.

If an authentication information object with this name already exists, REPLACE(*YES) must be specified.

The possible values are:

authentication-information-name
Specify the name of the authentication information object. The maximum string length is 48 characters.

Message Queue Manager name (MQMNAME)

The name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of an existing message queue manager. The maximum string length is 48 characters.

AuthInfo type (AUTHTYPE)

The type of the authentication information object. There is no default value

The possible values are:

***CRLLDAP**
The type of the authentication information object is CRLLDAP.

Connection name (CONNAME)

The DNS name or IP address of the host on which the LDAP server is running, together with an optional port number. The default port number is 389. No default is provided for the DNS name or IP address.

The possible values are:

***SAME**
The connection name remains unchanged from the original authentication information object.

connection-name
Specify the fully qualified DNS name or IP address of the host together with an optional port number. The maximum string length is 264 characters.

Replace (REPLACE)

Specifies whether the new authentication information object should replace an existing authentication information object with the same name.

The possible values are:

***NO** This definition does not replace any existing authentication information object with the same name. The command fails if the named authentication information object already exists.

***YES** Replace an existing authentication information object. A new object is created if the named authentication information object does not exist.

Text 'description' (TEXT)

A short text description of the authentication information object.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SAME**
The text string is unchanged.

***NONE**
The text is set to a blank string.

description
The string length can be up to 64 characters enclosed in apostrophes.

User name (USERNAME)

The distinguished name of the user that is binding to the directory. The default user name is blank.

The possible values are:

***SAME**
The user name is unchanged.

***NONE**
The user name is blank.

LDAP-user-name
Specify the distinguished name of the LDAP user. The maximum string length is 1024 characters.

User password (PASSWORD)

The password for the LDAP user.

The possible values are:

***SAME**
The password is unchanged.

***NONE**
The password is blank.

LDAP-password
The LDAP user password. The maximum string length is 32 characters.

Examples

None

Error messages

Unknown

Copy MQ Channel (CPYMQMCHL)

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

The Copy MQ Channel (CPYMQMCHL) command creates a new MQ channel definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing channel definition.

Parameters

Keyword	Description	Choices	Notes
FROMCHL	From channel	*Character value*	Required, Key, Positional 1
TOCHL	To channel	*Character value*	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN	Optional, Key, Positional 4

Keyword	Description	Choices	Notes
REPLACE	Replace	***NO** , *YES	Optional, Positional 5
TRPTYPE	Transport type	*LU62, *TCP, ***SAME**	Optional, Positional 6
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 7
TGTMQMNAME	Target Queue Manager	*Character value*, *NONE, ***SAME**	Optional, Positional 8
CONNAME	Connection name	*Character value*, *NONE, ***SAME**	Optional, Positional 9
TPNAME	Transaction Program Name	*Character value*, *BLANK, ***SAME**	Optional, Positional 10
MODENAME	Mode Name	*Character value*, *BLANK, ***SAME**	Optional, Positional 11
TMQNAME	Transmission queue	*Character value*, ***SAME**	Optional, Positional 12
MCANAME	Message channel agent	Single values: ***SAME** , *NONE Other values: *Qualified object name*	Optional, Positional 13
	Qualifier 1: Message channel agent	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MCAUSRID	Message channel agent user ID	*Character value*, *NONE, *PUBLIC, ***SAME**	Optional, Positional 14
MCATYPE	Message channel agent Type	*PROCESS, *THREAD, ***SAME**	Optional, Positional 15
BATCHINT	Batch Interval	0-999999999, ***SAME**	Optional, Positional 16
BATCHSIZE	Batch size	1-9999, ***SAME**	Optional, Positional 17
DSCITV	Disconnect interval	0-999999, ***SAME**	Optional, Positional 18
SHORTTMR	Short retry interval	0-999999999, ***SAME**	Optional, Positional 19

Keyword	Description	Choices	Notes
SHORTRTY	Short retry count	0-999999999, ***SAME**	Optional, Positional 20
LONGTMR	Long retry interval	0-999999999, ***SAME**	Optional, Positional 21
LONGRTY	Long retry count	0-999999999, ***SAME**	Optional, Positional 22
SCYEXIT	Security exit	Single values: ***SAME** , *NONE Other values: *Qualified object name*	Optional, Positional 23
	Qualifier 1: Security exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CSCYEXIT	Security exit	*Character value*, ***SAME** , *NONE	Optional, Positional 24
SCYUSRDATA	Security exit user data	*Character value*, ***SAME** , *NONE	Optional, Positional 25
SNDEXIT	Send exit	Single values: ***SAME** , *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 26
	Qualifier 1: Send exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CSNDEXIT	Send exit	Single values: ***SAME** , *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 27
SNDUSRDATA	Send exit user data	Values (up to 10 repetitions): *Character value*, ***SAME** , *NONE	Optional, Positional 28
RCVEXIT	Receive exit	Single values: ***SAME** , *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 29
	Qualifier 1: Receive exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CRCVEXIT	Receive exit	Single values: ***SAME** , *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 30
RCVUSRDATA	Receive exit user data	Values (up to 10 repetitions): *Character value*, ***SAME** , *NONE	Optional, Positional 31

Keyword	Description	Choices	Notes
MSGEXIT	Message exit	Single values: ***SAME** , *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 32
	Qualifier 1: Message exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGUSRDATA	Message exit user data	Values (up to 10 repetitions): *Character value*, ***SAME** , *NONE	Optional, Positional 33
MSGRTYEXIT	Message retry exit	Single values: ***SAME** , *NONE Other values: *Qualified object name*	Optional, Positional 34
	Qualifier 1: Message retry exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGRTYDATA	Message retry exit data	*Character value*, ***SAME** , *NONE	Optional, Positional 35
MSGRTYNBR	Number of message retries	0-999999999, ***SAME**	Optional, Positional 36
MSGRTYITV	Message retry interval	0-999999999, ***SAME**	Optional, Positional 37
CVTMSG	Convert message	*YES, *NO, ***SAME**	Optional, Positional 38
PUTAUT	Put authority	*DFT, *CTX, ***SAME**	Optional, Positional 39
SEQNUMWRAP	Sequence number wrap	100-999999999, ***SAME**	Optional, Positional 40
MAXMSGLEN	Maximum message length	0-104857600, ***SAME**	Optional, Positional 41
HRTBTINTVL	Heartbeat interval	0-999999999, ***SAME**	Optional, Positional 42
NPMSPEED	Non Persistent Message Speed	*FAST, *NORMAL, ***SAME**	Optional, Positional 43
CLUSTER	Cluster Name	*Character value*, *NONE, ***SAME**	Optional, Positional 44
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SAME**	Optional, Positional 45

Keyword	Description	Choices	Notes
NETPRTY	Network Connection Priority	0-9, ***SAME**	Optional, Positional 46
SSLCIPH	SSL CipherSpec	*Character value*, *NULL_MD5, *NULL_SHA, *RC4_MD5_EXPORT, *RC4_MD5_US, *RC4_SHA_US, *RC2_MD5_EXPORT, *DES_SHA_EXPORT, *TRIPLE_DES_SHA_US, *AES_SHA_US, *TLS_RSA_WITH_NULL_MD5, *TLS_RSA_WITH_NULL_SHA, *TLS_RSA_EXPORT_WITH_RC4_40_MD5, *TLS_RSA_WITH_RC4_128_MD5, *TLS_RSA_WITH_RC4_128_SHA, *TLS_RSA_EXPORT_WITH_RC2_40_MD5, *TLS_RSA_WITH_DES_CBC_SHA, *TLS_RSA_WITH_3DES_EDE_CBC_SHA, *TLS_RSA_WITH_AES_128_CBC_SHA, *TLS_RSA_WITH_AES_256_CBC_SHA, *NONE, ***SAME**	Optional, Positional 47
SSLCAUTH	SSL Client Authentication	*REQUIRED, *OPTIONAL, ***SAME**	Optional, Positional 48
SSLPEER	SSL Peer name	*Character value*, *NONE, ***SAME**	Optional, Positional 49
LOCLADDR	Local communication address	*Character value*, *NONE, ***SAME**	Optional, Positional 50
BATCHHB	Batch Heartbeat Interval	0-999999999, ***SAME**	Optional, Positional 51
USERID	Task user identifier	*Character value*, *NONE, ***SAME**	Optional, Positional 52
PASSWORD	Password	*Character value*, *NONE, ***SAME**	Optional, Positional 53
KAINT	Keep Alive Interval	0-99999, ***SAME** , *AUTO	Optional, Positional 54
COMPHDR	Header Compression	Values (up to 2 repetitions): *NONE, *SYSTEM, ***SAME**	Optional, Positional 55
COMPMSG	Message Compression	Single values: *ANY Other values (up to 4 repetitions): *NONE, *RLE, *ZLIBHIGH, *ZLIBFAST, ***SAME**	Optional, Positional 56
MONCHL	Channel Monitoring	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SAME**	Optional, Positional 57
STATCHL	Channel Statistics	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SAME**	Optional, Positional 58

Keyword	Description	Choices	Notes
CLWLRANK	Cluster Workload Rank	0-9, ***SAME**	Optional, Positional 59
CLWLPRTY	Cluster Workload Priority	0-9, ***SAME**	Optional, Positional 60
CLWLWGHT	Cluster Channel Weight	1-99, ***SAME**	Optional, Positional 61
SHARECNV	Sharing Conversations	0-999999999, ***SAME**	Optional, Positional 62
PROPCTL	Property Control	*COMPAT, *NONE, *ALL, ***SAME**	Optional, Positional 63
MAXINST	Maximum Instances	0-999999999, ***SAME**	Optional, Positional 64
MAXINSTC	Maximum Instances Per Client	0-999999999, ***SAME**	Optional, Positional 65
CLNTWGHT	Client Channel Weight	0-99, ***SAME**	Optional, Positional 66
AFFINITY	Client Channel Affinity	*PREFERRED, *NONE, ***SAME**	Optional, Positional 67

From channel (FROMCHL)

Specifies the name of the existing channel definition that contains values for the attributes that are not specified in this command.

The possible values are:

from-channel-name
Specify the name of the source MQ channel.

To channel (TOCHL)

Specifies the name of the new channel definition. The name can contain a maximum of 20 characters. Channel names must be unique. If a channel definition with this name already exists, REPLACE(*YES) must be specified.

The possible values are:

to-channel-name
Specify the name of MQ channel being created.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Channel type (CHLTYPE)

Specifies the type of the channel being copied.

The possible values are:

***SDR** Sender channel

***SVR** Server channel

***RCVR**
Receiver channel

***RQSTR**
Requester channel

***SVRCN**
Server-connection channel

***CLUSSDR**
Cluster-sender channel

***CLUSRCVR**
Cluster-receiver channel

***CLTCN**
Client-connection channel

Replace (REPLACE)

Specifies whether the new channel definition replaces an existing channel definition with the same name.

The possible values are:

***NO** Do not replace the existing channel definition. The command fails if the named channel definition already exists.

***YES** Replace the existing channel definition. If there is no definition with the same name a new definition is created.

Transport type (TRPTYPE)

Specifies the transmission protocol.

The possible values are:

***SAME**
The attribute is unchanged.

***LU62** SNA LU 6.2.

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

Text 'description' (TEXT)

Specifies text that briefly describes the channel definition.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Target Queue Manager (TGTMQMNAME)

Specifies the name of the target queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The name of the target queue manager for a client connection channel (CHLTYPE) *CLTCN is unspecified.

message-queue-manager-name
The name of the target message queue manager for a client connection channel (CHLTYPE) *CLTCN.

For other channel types this parameter must not be specified.

Connection name (CONNAME)

Specifies the name of the machine to connect.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The connection name is blank.

connection-name
Specify the connection name as required by the transmission protocol:

- For *LU62, specify the name of the CSI object.
- For *TCP, specify either the host name, or the network address of the remote machine (or the local machine for cluster-receiver channels). This can be followed by an optional port number enclosed in parentheses.

  If a connection name is not specified for cluster-receiver channels a connection name is automatically generated, assuming the default port and using the current IP address of the system.

  Where a port is not specified the default port 1414 is assumed.

For cluster-receiver channels the connection name relates to the local queue manager, and for other channels it relates to the target queue manager.

This parameter is required for channels with channel type (CHLTYPE) of *SDR, *RQSTR, *CLTCN and *CLUSSDR. It is optional for *SVR and *CLUSRCVR channels, and is not valid for *RCVR or *SVRCN channels.

Transaction Program Name (TPNAME)

This parameter is valid for channels with a TRPTYPE defined as LU 6.2 only.

This parameter must be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it must be set to blanks. The name is taken instead from the CPI-C Communications Side Object.

This parameter is not valid for channels with a CHLTYPE defined as *RCVR.

The possible values are:

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

***NONE**
No transaction program name is specified.

***BLANK**
The transaction program name is taken from CPI-C Communications Side Object. The side object name must be specified in the CONNAME parameter.

transaction-program-name
Specify the SNA transaction program name.

Mode Name (MODENAME)

This parameter is valid for channels with a TRPTYPE defined as LU 6.2. If TRPTYPE is not defined as LU 6.2 the data is ignored and no error message is issued.

If specified, the value must be set to the SNA mode name, unless the CONNAME contains a side-object name, in which case it must be set to blanks. The name is then taken from the CPI-C Communications Side Object.

This parameter is not valid for channels with CHLTYPE defined as *RCVR or *SVRCONN.

The possible values are:

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

***NONE**
No mode name is specified.

***BLANK**
Name will be taken from the CPI-C Communications Side Object. This must be specified in the CONNAME parameter.

SNA-mode-name
Specify the SNA Mode Name

Transmission queue (TMQNAME)

Specifies the name of the transmission queue.

The possible values are:

***SAME**
The attribute is unchanged.

transmission-queue-name
Specify the name of the transmission queue. A transmission queue name is required if the CHLTYPE is defined as *SDR or *SVR.

For other channel types this parameter must not be specified.

Message channel agent (MCANAME)

This parameter is reserved and should not be used.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The MCA program name is blank.

This parameter cannot be specified if the CHLTYPE is defined as *RCVR, *SVRCN, or *CLTCN.

Message channel agent user ID (MCAUSRID)

Specifies the message channel agent user identifier which is to be used by the message channel agent for authorization to access MQ resources, including (if PUTAUT is *DFT) authorization to put the message to the destination queue for receiver or requester channels.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The message channel agent uses its default user identifier.

***PUBLIC**
Uses the public authority.

mca-user-identifier
Specify the user identifier to be used.

This parameter cannot be specified for a channel type (CHLTYPE) of *CLTCN.

Message channel agent Type (MCATYPE)

Specifies whether the message channel agent program should run as a thread or as a process.

The possible values are:

***SAME**
The attribute is unchanged.

***PROCESS**
The message channel agent runs as a separate process.

***THREAD**
The message channel agent runs as a separate thread.

This parameter can only be specified for channels with CHLTYPE defined as *SDR, *SVR, *RQSTR, *CLUSSDR or *CLUSRCVR.

Batch Interval (BATCHINT)

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

The batch is terminated by which ever of the following occurs first: BATCHSZ messages have been sent, or the transmission queue is empty and BATCHINT is exceeded.

The default value is 0, which means that the batch is terminated as soon as the transmission queue becomes empty (or the BATCHSZ limit is reached).

The value must be in the range 0 through 999999999.

This parameter is valid for channels with CHLTYPE defined as *SDR, *SVR, *CLUSSDR, or *CLUSRCVR.

The possible values are:

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

batch-interval
Specify a value ranging from 0 through 999999999

Batch size (BATCHSIZE)

Specifies the maximum number of messages that can be sent down a channel before a checkpoint is taken.

The possible values are:

***SAME**
The attribute is unchanged.

batch-size
Specify a value ranging from 1 through 9999.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Disconnect interval (DSCITV)

Specifies the disconnect interval, which defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before closing the channel.

The possible values are:

***SAME**
The attribute is unchanged.

disconnect-interval
Specify a value ranging from 0 through 999999.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR or *CLTCN.

Short retry interval (SHORTTMR)

Specifies the short retry wait interval for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. This defines the interval between attempts to establish a connection to the remote machine.

The possible values are:

***SAME**
The attribute is unchanged.

short-retry-interval
Specify a value ranging from 0 through 999999999.

Short retry count (SHORTRTY)

Specifies the short retry count for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of attempts that are made to establish a connection to the remote machine, at intervals specified by SHORTTMR, before the (normally longer) LONGRTY and LONGTMR are used.

The possible values are:

***SAME**
The attribute is unchanged.

short-retry-count
Specify a value ranging from 0 through 999999999. A value of 0 means that no retries are allowed.

Long retry interval (LONGTMR)

Specifies the long retry wait interval for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by SHORTRTY has been exhausted.

The possible values are:

***SAME**
The attribute is unchanged.

long-retry-interval
Specify a value in the range 0 through 999999999.

Note: For implementation reasons, the maximum retry interval that can be used is 999999; values exceeding this are treated as 999999.

Long retry count (LONGRTY)

Specifies the long retry count for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of further attempts that are made to connect to the remote machine, at intervals specified by LONGTMR, after the count specified by SHORTRTY has been exhausted. An error message is logged if the connection is not established after the defined number of attempts.

The possible values are:

***SAME**
The attribute is unchanged.

long-retry-count
Specify a value in the range 0 through 999999999. A value of 0 means that no retries are allowed.

Security exit (SCYEXIT)

Specifies the name of the program to be called as the security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

  Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.
- On receipt of a response to a security message flow.

  Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The security exit program is not invoked.

security-exit-name
Specify the name of the security exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

Security exit (CSCYEXIT)

Specifies the name of the program to be called as the client security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

  Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.
- On receipt of a response to a security message flow.

  Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The client security exit program is not invoked.

security-exit-name
Specify the name of the client security exit program.

Security exit user data (SCYUSRDATA)

Specifies a maximum of 32 characters of user data that is passed to the security exit program.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data for the security exit program is not specified.

security-exit-user-data
Specify the user data for the security exit.

Send exit (SNDEXIT)

Specifies the entry point of the program to be called as the send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The send exit program is not invoked.

send-exit-name
Specify the name of the send exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

Send exit (CSNDEXIT)

Specifies the entry point of the program to be called as the client send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The client send exit program is not invoked.

send-exit-name
Specify the name of the client send exit program.

Send exit user data (SNDUSRDATA)

Specifies a maximum of 32 characters of user data that is passed to the send exit program.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data for the send exit program is not specified.

send-exit-user-data
Specify the user data for the send exit program.

Receive exit (RCVEXIT)

Specifies the entry point of the program to be called as the receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The receive exit program is not invoked.

receive-exit-name
Specify the name of the receive exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

Receive exit (CRCVEXIT)

Specifies the entry point of the program to be called as the client receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The client receive exit program is not invoked.

receive-exit-name
Specify the name of the client receive exit program.

Receive exit user data (RCVUSRDATA)

Specifies a maximum of 32 characters of user data that is passed to the receive exit program.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data for the receive exit program is not specified.

receive-exit-user-data
Specify a maximum of 32 characters of user data for the receive exit.

Message exit (MSGEXIT)

Specifies the entry point of the program to be called as the message exit. If a nonblank name is defined, the exit is invoked immediately after a message has been retrieved from the transmission queue. The exit is given the entire application message and message descriptor for modification.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The message exit program is not invoked.

message-exit-name
Specify the name of the message exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Message exit user data (MSGUSRDATA)

Specifies user data that is passed to the message exit program.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data for the message exit program is not specified.

message-exit-user-data
Specify a maximum of 32 characters of user data that is passed to the message exit program.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Message retry exit (MSGRTYEXIT)

Specifies the entry point of the program to be called as the message retry exit.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The message retry exit program is not invoked.

message-retry-exit-name
Specify the name of the message retry exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Message retry exit data (MSGRTYDATA)

Specifies user data that is passed to the message retry exit program.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data for the message retry exit program is not specified.

message-retry-exit-user-data
Specify a maximum of 32 characters of user data that is passed to the message retry exit program.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Number of message retries (MSGRTYNBR)

Specifies the number of times the channel will retry before it decides it cannot deliver the message.

This parameter is used by the channel as an alternative to a message retry exit when MSGRTYEXIT is defined as *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

message-retry-number
Specify a value ranging from 0 through 999999999. A value of 0 indicates no retries will be performed.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Message retry interval (MSGRTYITV)

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

This parameter is used by the channel as an alternative to a message retry exit when MSGRTYEXIT is defined as *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

message-retry-number
Specify a value ranging from 0 through 999999999. A value of 0 indicates that the retry will be performed as soon as possible.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Convert message (CVTMSG)

Specifies whether the application data in the message should be converted before the message is transmitted.

The possible values are:

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

***YES** The application data in the message is converted before sending.

***NO** The application data in the message is not converted before sending.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR, *CLTCN or *SVRCN.

Put authority (PUTAUT)

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

The possible values are:

***SAME**
The attribute is unchanged.

***DFT** No authority check is made before the message is put on the destination queue.

***CTX** The user identifier in the message context information is used to establish authority to put the message.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Sequence number wrap (SEQNUMWRAP)

Specifies the maximum message sequence number. When the maximum is reached, sequence numbers wrap to start again at 1.

Note: The maximum message sequence number is not negotiable; the local and remote channels must wrap at the same number.

The possible values are:

***SAME**
The attribute is unchanged.

sequence-number-wrap-value
Specify a value ranging from 100 through 999999999.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Maximum message length (MAXMSGLEN)

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

The possible values are:

***SAME**
The attribute is unchanged.

maximum-message-length
Specify a value ranging from 0 through 104857600. A value of 0 indicates that the maximum length is unlimited.

Heartbeat interval (HRTBTINTVL)

Specifies the time, in seconds, between heartbeat flows passed from the sending MCA when there are no messages on the transmission queue. The heartbeat exchange gives the receiving MCA the opportunity to quiesce the channel. This applies only to sender, server, cluster sender and cluster receiver (*SDR, *SVR, *CLUSSDR and *CLUSRCVR) channels.

The possible values are:

***SAME**
The attribute is unchanged.

heart-beat-interval
Specify a value ranging from 0 through 999999999. A value of 0 means that no heartbeat exchanges are to take place.

Non Persistent Message Speed (NPMSPEED)

Specifies whether the channel supports fast non persistant messages.

The possible values are:

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

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

***NORMAL**
The channel does not support fast non persistant messages.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Cluster Name (CLUSTER)

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

This parameter is valid only for *CLUSSDR and *CLUSRCVR channels. If the CLUSNL parameter is non-blank, this parameter must be blank.

The possible values are:

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

***NONE**
No cluster name is specififed.

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

Cluster Name List (CLUSNL)

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

This parameter is valid only for *CLUSSDR and *CLUSRCVR channels. If the CLUSTER parameter is non-blank, this parameter must be blank.

The possible values are:

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

***NONE**
No cluster namelist is specififed.

cluster-name-list
The name of the namelist specifying a list of clusters to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

Network Connection Priority (NETPRTY)

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

This parameter is valid only for *CLUSRCVR channels.

The possible values are:

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

network-connection-priority
Specify a value ranging from 0 through 9 where 0 is the lowest priority.

SSL CipherSpec (SSLCIPH)

SSLCIPH specifies the CipherSpec used in SSL channel negotiation. The possible values are:

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

cipherspec
The name of the CipherSpec.

SSL Client Authentication (SSLCAUTH)

SSLCAUTH specifies whether the channel carries out client authentication over SSL. The parameter is used only for channels with SSLCIPH specified.

The possible values are:

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

***REQUIRED**
Client authentication is required.

***OPTIONAL**
Client authentication is optional.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *CLTCN or *CLUSSDR.

SSL Peer name (SSLPEER)

SSLPEER specifies the X500 peer name used in SSL channel negotiation. The possible values are:

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

x500peername
The X500 peer name to use.

Local communication address (LOCLADDR)

Specifies the local communication address for the channel.

This parameter is only valid for *SDR, *SVR, *RQSTR, *CLUSSDR, *CLUSRCVR and *CLTCN channels.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The connection is blank.

local-address
Only valid for transport type TCP/IP. Specify the optional IP address and optional port or port range used for outbound TCP/IP communications. The format is LOCLADDR([ip-addr][(low-port[,high-port])]).

Batch Heartbeat Interval (BATCHHB)

The time in milliseconds used to determine whether batch heartbeating occurs on this channel. Batch heartbeating allows channels to determine whether the remote channel instance is still active before going indoubt. A batch heartbeat will occur if a channel MCA has not communicated with the remote channel within the specified time.

The possible values are:

***SAME**
The attribute is unchanged.

batch-heartbeat-interval
Specify a value ranging from 0 through 999999999. A value of 0 indicates that batch heartbeating is not to be used.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR, *CLTCN or *SVRCN.

Task user identifier (USERID)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

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

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

The possible values are:

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

***NONE**
No user identifier is specified.

user-identifier
Specify the task user identifier.

Password (PASSWORD)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

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

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

The possible values are:

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

***NONE**
No password is specified.

password
Specify the password.

Keep Alive Interval (KAINT)

Specifies the keep alive timing interval for this channel.

The possible values are:

***SAME**
The attribute is unchanged.

***AUTO**
The keep alive interval is calculated based upon the negotiated heartbeat value as follows:

- If the negotiated HBINT is greater than 0, keep alive interval is set to that value plus 60 seconds.
- If the negotiated HBINT is 0, the value used is that specified by the KEEPALIVEOPTIONS statement in the TCP profile configuration data set.

keep-alive-interval
Specify a value ranging from 0 through 99999.

Header Compression (COMPHDR)

The list of header data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (*SDR, *SVR, *CLUSSDR, *CLUSRCVR and *CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No header data compression is performed.

***SYSTEM**
Header data compression is performed.

Message Compression (COMPMSG)

The list of message data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (*SDR, *SVR, *CLUSSDR, *CLUSRCVR and *CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No message data compression is performed.

***RLE** Message data compression is performed using run-length encoding.

***ZLIBFAST**
Message data compression is performed using the zlib compression technique. A fast compression time is preferred.

***ZLIBHIGH**
Message data compression is performed using the zlib compression technique. A high level of compression is preferred.

***ANY** Any compression technique supported by the queue manager can be used. This option is only valid for channel types receiver, requester and server connection (*RCVR, *RQSTR and *SVRCN).

Channel Monitoring (MONCHL)

Controls the collection of online monitoring data.

Online monitoring data is not collected when the queue manager attribute MONCHL is set to *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
The collection of online monitoring data is inherited from the setting of the queue manager attribute MONCHL.

***OFF** Online Monitoring Data collection for this channel is switched off.

***LOW** Monitoring data collection is turned on with a low ratio of data collection.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for a channel type (CHLTYPE) of *CLTCN.

Channel Statistics (STATCHL)

Controls the collection of statistics data.

Statistics data is not collected when the queue manager attribute STATCHL is set to *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
Statistics data collection is based upon the setting of the queue manager attribute STATCHL.

***OFF** Statistics data collection for this channel is switched off.

***LOW** Statistics data collection is turned on with a low ratio of data collection.

***MEDIUM**
Statistics data collection is turned on with a moderate ratio of data collection.

***HIGH**
Statistics data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Cluster Workload Rank (CLWLRANK)

Specifies the cluster workload rank of the channel.

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-rank
The cluster workload rank of the channel in the range 0 through 9.

Cluster Workload Priority (CLWLPRTY)

Specifies the cluster workload priority of the channel.

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-priority
The cluster workload priority of the channel in the range 0 through 9.

Cluster Channel Weight (CLWLWGHT)

Specifies the cluster workload weight of the channel.

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-weight
The cluster workload weight of the channel in the range 1 through 99.

Sharing Conversations (SHARECNV)

Specifies the maximum the number of conversations which can be shared over a particular TCP/IP client channel instance (socket).

This parameter is valid for channels with CHLTYPE defined as *CLTCN or *SVRCN.

The possible values are:

***SAME**
The attribute is unchanged.

0 Specifies no sharing of conversations over a TCP/IP socket. The channel instance runs in a mode prior to that of WebSphere MQ Version 7.0, with regard to:

- Administrator stop-quiesce
- Heartbeating
- Read ahead

1 Specifies no sharing of conversations over a TCP/IP socket. Client heartbeating and read ahead are available, whether in an MQGET call or not, and channel quiescing is more controllable.

shared-conversations
The number of shared conversations in the range 2 through 999999999.

This parameter is only valid for client-connection and server-connection channels.

Note: If the client-connection SHARECNV value does not match the server-connection SHARECNV value, the lower of the two values is used.

Property Control (PROPCTL)

Specifies what happens to properties of messages when the message is about to be sent to a V6 or prior queue manager (a queue manager that does not understand the concept of a property descriptor).

The possible values are:

***SAME**
The attribute is unchanged.

***COMPAT**
If the message contains a property with a prefix of "mcd.", "jms.", "usr." or "mqext." then all optional message properties, except those in the message descriptor (or extension) will be placed in one or more MQRFH2 headers in the message data before the message is sent to the remote queue manager.

***NONE**
All properties of the message, except those in the message descriptor (or extension), will be removed from the message before the message is sent to the remote queue manager.

***ALL** All properties of the message will be included with the message when it is sent to the remote queue manager. The properties, except those in the message descriptor (or extension), will be placed in one or more MQRFH2 headers in the message data.

Maximum Instances (MAXINST)

Specifies the maximum number of simultaneous instances of an individual server-connection channel.

The possible values are:

***SAME**
The attribute is unchanged.

maximum-instances
The maximum number of simultaneous instances of the channel in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

Maximum Instances Per Client (MAXINSTC)

Specifies the maximum number of simultaneous instances of an individual server-connection channel which can be started from a single client.

The possible values are:

***SAME**
The attribute is unchanged.

maximum-instances-per-client
The maximum number of simultaneous instances of the channel which can be in the started from a single client in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running from individual clients, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

Client Channel Weight (CLNTWGHT)

The client channel weighting attribute is used so client channel definitions can be selected at random based on their weighting when more than one suitable definition is available.

The possible values are:

***SAME**
The attribute is unchanged.

client-channel-weight
The client channel weight in the range 0 through 99.

Client Channel Affinity (AFFINITY)

The channel affinity attribute is used so client applications that connect multiple times using the same queue manager name can choose whether to use the same client channel definition for each connection.

The possible values are:

***SAME**
The attribute is unchanged.

***PREFERRED**
The first connection in a process reading a CCDT creates a list of applicable definitions based on the weighting with any applicable CLNTWGHT(0) definitions first and in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful non CLNTWGHT(0) definitions are moved to the end of the list. CLNTWGHT(0) definitions remain at the start of the list and are selected first for each connection.

***NONE**
The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the weighting with any applicable CLNTWGHT(0) definitions selected first in alphabetical order.

Examples

None

Error messages

Unknown

Copy MQ Listener (CPYMQMLSR)

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

The Copy MQ Listener (CPYMQMLSR) command creates an MQ listener definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing listener definition.

Parameters

Keyword	Description	Choices	Notes
FROMLSR	From Listener	*Character value*	Required, Key, Positional 1
TOLSR	To Listener	*Character value*	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO**, *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
CONTROL	Listener control	***SAME**, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 6
PORT	Port number	0-65535, ***SAME**	Optional, Positional 7
IPADDR	IP Address	*Character value*, *BLANK, ***SAME**	Optional, Positional 8
BACKLOG	Listener backlog	0-999999999, ***SAME**	Optional, Positional 9

From Listener (FROMLSR)

Specifies the name of the existing listener definition to provide values for the attributes not specified in this command.

The possible values are:

from-listener-name
Specify the name of the source MQ listener.

To Listener (TOLSR)

Specifies the name of the new listener definition to be created. The name can contain a maximum of 48 characters.

If a listener definition with this name already exists, REPLACE(*YES) must be specified.

The possible values are:

to-listener-name
Specify the name of the new listener being created.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Replace (REPLACE)

Specifies whether the new listener definition will replace an existing listener definition with the same name.

The possible values are:

***NO** This definition does not replace any existing listener definition with the same name. The command fails if the named listener definition already exists.

***YES** Replace the existing listener definition. If there is no definition with the same name, a new definition is created.

Text 'description' (TEXT)

Specifies text that briefly describes the listener definition.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Listener control (CONTROL)

Whether the listener starts automatically when the queue manager is started.

The possible values are:

***SAME**
The attribute is unchanged.

***MANUAL**
The listener is not automatically started or stopped.

***QMGR**
The listener is started and stopped as the queue manager is started and stopped.

***STARTONLY**
The listener is started as the queue manager is started, but is not automatically stopped when the queue manager is stopped.

Port number (PORT)

The port number to be used by the listener.

The possible values are:

***SAME**
The attribute is unchanged.

port-number
The port number to be used.

IP Address (IPADDR)

The IP address to be used by the listener.

The possible values are:

***SAME**
The attribute is unchanged.

ip-addr
The IP address to be used.

Listener backlog (BACKLOG)

The number of concurrent connection requests the listener supports.

The possible values are:

***SAME**
The attribute is unchanged.

backlog
The number of concurrent connection requests supported.

Examples

None

Error messages

Unknown

Copy MQ Namelist (CPYMQMNL)

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

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

Parameters

Keyword	Description	Choices	Notes
FROMNL	From Namelist	*Character value*	Required, Key, Positional 1
TONL	To Namelist	*Character value*	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO**, *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
NAMES	List of Names	Values (up to 256 repetitions): *Character value*, *BLANKS, ***SAME**, *NONE	Optional, Positional 6

From Namelist (FROMNL)

Specifies the name of the existing namelist, to provide values for the attributes not specified in this command.

from-namelist
Specify the name of the source namelist.

To Namelist (TONL)

The name of the new namelist to be created. The name can contain a maximum of 48 characters.

If a namelist with this name already exists, REPLACE(*YES) must be specified.

to-namelist
Specify the name of the MQ namelist being created.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used.

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

Replace (REPLACE)

Specifies whether the new namelist should replace an existing namelist with the same name.

***NO** Do not replace the existing namelist. The command fails if the named namelist already exists.

***YES** Replace the existing namelist. If there is no namelist with the same name, a new namelist is created.

Text 'description' (TEXT)

Specifies text that briefly describes the namelist.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

***SAME**
The attribute is unchanged.

description
Specify no more than 64 characters enclosed in apostrophes.

List of Names (NAMES)

List of names. This is the list of names to be created. The names can be of any type, but must conform to the rules for naming MQ objects.

***SAME**
The attribute is unchanged.

namelist
The list to create. An empty list is valid.

Examples

None

Error messages

Unknown

Copy MQ Process (CPYMQMPRC)

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

The Copy MQ Process (CPYMQMPRC) command creates an MQ process definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing process definition.

Parameters

Keyword	Description	Choices	Notes
FROMPRC	From process	*Character value*	Required, Key, Positional 1
TOPRC	To process	*Character value*	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO** , *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
APPTYPE	Application type	*Integer*, ***SAME** , *CICS, *MVS, *IMS, *OS2, *DOS, *UNIX, *QMGR, *OS400, *WINDOWS, *CICS_VSE, *WINDOWS_NT, *VMS, *NSK, *VOS, *IMS_BRIDGE, *XCF, *CICS_BRIDGE, *NOTES_AGENT, *BROKER, *JAVA, *DQM	Optional, Positional 6

Keyword	Description	Choices	Notes
APPID	Application identifier	*Character value*, ***SAME**	Optional, Positional 7
USRDATA	User data	*Character value*, ***SAME** , *NONE	Optional, Positional 8
ENVDATA	Environment data	*Character value*, ***SAME** , *NONE	Optional, Positional 9

From process (FROMPRC)

Specifies the name of the existing process definition to provide values for the attributes not specified in this command.

The possible values are:

from-process-name
Specify the name of the source MQ process.

To process (TOPRC)

The name of the new process definition to be created. The name can contain a maximum of 48 characters.

If a process definition with this name already exists, REPLACE(*YES) must be specified.

The possible values are:

to-process-name
Specify the name of the MQ process being created.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Replace (REPLACE)

Specifies whether the new process definition should replace an existing process definition with the same name.

The possible values are:

***NO** This definition does not replace any existing process definition with the same name. The command fails if the named process definition already exists.

***YES** Replace the existing process definition. If there is no definition with the same name, a new definition is created.

Text 'description' (TEXT)

Specifies text that briefly describes the process definition.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Application type (APPTYPE)

The type of application started.

The possible values are:

***SAME**
The attribute is unchanged.

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

***MVS** Represents an MVS application.

***IMS** Represents an IMS application.

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

***DOS** Represents a DOS application.

***UNIX**
Represents a UNIX application.

***QMGR**
Represents a queue manager.

***OS400**
Represents an i5/OS application.

***WINDOWS**
Represents a Windows application.

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

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

***VMS** Represents a VMS application.

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

***VOS** Represents a VOS application.

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

***XCF** Represents an XCF application.

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

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

***BROKER**
Represents a broker application.

***JAVA** Represents a Java application.

***DQM**
Represents a DQM application.

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

Application identifier (APPID)

Application identifier. This is the name of the application to be started, on the platform for which the command is processing. It is typically a program name and library name.

The possible values are:

***SAME**
The attribute is unchanged.

application-id
The maximum length is 256 characters.

User data (USRDATA)

A character string that contains user information pertaining to the application, as defined by APPID, to start.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The user data is blank.

user-data
Specify up to 128 characters of user data.

Environment data (ENVDATA)

A character string that contains environment information pertaining to the application, as defined by APPID, to start.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The environment data is blank.

environment-data
The maximum length is 128 characters.

Examples

None

Error messages

Unknown

Copy MQ Queue (CPYMQMQ)

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

The Copy MQ Queue (CPYMQMQ) command creates a queue definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing queue definition.

Parameters

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

Keyword	Description	Choices	Notes
INITQNAME	Initiation queue	*Character value*, *NONE, ***SAME**	Optional, Positional 26
USAGE	Usage	***SAME** , *NORMAL, *TMQ	Optional, Positional 27
DFNTYPE	Definition type	***SAME** , *TEMPDYN, *PERMDYN	Optional, Positional 28
TGTQNAME	Target queue	*Character value*, ***SAME**	Optional, Positional 29
RMTQNAME	Remote queue	*Character value*, ***SAME** , *NONE	Optional, Positional 30
RMTMQMNAME	Remote Message Queue Manager	*Character value*, ***SAME**	Optional, Positional 31
TMQNAME	Transmission queue	*Character value*, *NONE, ***SAME**	Optional, Positional 32
HIGHTHLD	Queue depth high threshold	0-100, ***SAME**	Optional, Positional 33
LOWTHLD	Queue depth low threshold	0-100, ***SAME**	Optional, Positional 34
FULLEVT	Queue full events enabled	***SAME** , *NO, *YES	Optional, Positional 35
HIGHEVT	Queue high events enabled	***SAME** , *NO, *YES	Optional, Positional 36
LOWEVT	Queue low events enabled	***SAME** , *NO, *YES	Optional, Positional 37
SRVITV	Service interval	0-999999999, ***SAME**	Optional, Positional 38
SRVEVT	Service interval events	***SAME** , *HIGH, *OK, *NONE	Optional, Positional 39
DISTLIST	Distribution list support	***SAME** , *NO, *YES	Optional, Positional 40
CLUSTER	Cluster Name	*Character value*, ***SAME** , *NONE	Optional, Positional 41
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SAME**	Optional, Positional 42
DEFBIND	Default Binding	***SAME** , *OPEN, *NOTFIXED	Optional, Positional 43
CLWLRANK	Cluster Workload Rank	0-9, ***SAME**	Optional, Positional 44
CLWLPRTY	Cluster Workload Priority	0-9, ***SAME**	Optional, Positional 45
CLWLUSEQ	Cluster workload queue use	***SAME** , *QMGR, *LOCAL, *ANY	Optional, Positional 46
MONQ	Queue Monitoring	***SAME** , *QMGR, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 47
STATQ	Queue Statistics	***SAME** , *QMGR, *OFF, *ON	Optional, Positional 48

Keyword	Description	Choices	Notes
ACCTQ	Queue Accounting	***SAME** , *QMGR, *OFF, *ON	Optional, Positional 49
NPMCLASS	Non Persistent Message Class	***SAME** , *NORMAL, *HIGH	Optional, Positional 50
MSGREADAHD	Message Read Ahead	***SAME** , *DISABLED, *NO, *YES	Optional, Positional 51
DFTPUTRESP	Default Put Response	***SAME** , *SYNC, *ASYNC	Optional, Positional 52
PROPCTL	Property Control	***SAME** , *COMPAT, *NONE, *ALL, *FORCE	Optional, Positional 53
TARGTYPE	Target Type	***SAME** , *QUEUE, *TOPIC	Optional, Positional 54

From queue name (FROMQ)

Specifies the name of the existing queue definition, to provide values for the attributes not specified in this command.

The possible values are:

from-queue-name
Specify the name of the source queue.

To queue name (TOQ)

Specifies the name of the new queue definition. The name can contain a maximum of 48 characters. Queue name and type combinations must be unique; if a queue definition already exists with the name and type of the new queue, REPLACE(*YES) must be specified.

Note: The field length is 48 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

The possible values are:

to-queue-name
Specify the name of the queue being created.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Queue type (QTYPE)

Specifies the type of queue that is to be copied.

The possible values are:

***ALS** An alias queue.

***LCL** A local queue.

***RMT** A remote queue.

***MDL** A model queue.

Replace (REPLACE)

Specifies whether the new queue will replace an existing queue definition with the same name and type.

The possible values are:

***NO** Do not replace the existing queue definition. The command fails if the named queue already exists.

***YES** Replace the existing queue definition with the attributes of the FROMQ and the specified attributes.

The command fails if an application has the queue open or the USAGE attribute is changed.

Note: If the queue is a local queue, and a queue with the same name already exists, any messages already on that queue are retained.

Text 'description' (TEXT)

Specifies text that briefly describes the object.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Put enabled (PUTENBL)

Specifies whether messages can be put on the queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Messages cannot be added to the queue.

***YES** Messages can be added to the queue by authorized applications.

Default message priority (DFTPTY)

Specifies the default priority of messages put on the queue.

The possible values are:

***SAME**
The attribute is unchanged.

priority-value
Specify a value ranging from 0 through 9, where 9 is the highest priority.

Default message persistence (DFTMSGPST)

Specifies the default for message-persistence on the queue. Message persistence determines whether or not messages are preserved across restarts of the queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** By default, messages are lost across a restart of the queue manager.

***YES** By default, messages are preserved across a restart of the queue manager.

Process name (PRCNAME)

Specifies the local name of the MQ process that identifies the application that should be started when a trigger event occurs.

The process does not have to be available when the queue is created, but it must be available for a trigger event to occur.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
The process name is blank.

process-name
Specify the name of the MQ process.

Triggering enabled (TRGENBL)

Specifies whether trigger messages are written to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Triggering is not enabled. Trigger messages are not written to the initiation queue.

***YES** Triggering is enabled. Trigger messages are written to the initiation queue.

Get enabled (GETENBL)

Specifies whether applications are to be permitted to get messages from this queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Applications cannot retrieve messages from the queue.

***YES** Suitably authorized applications can retrieve messages from the queue.

Sharing enabled (SHARE)

Specifies whether multiple instances of applications can open this queue for input simultaneously.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Only a single application instance can open the queue for input.

***YES** More than one application instance can open the queue for input.

Default share option (DFTSHARE)

Specifies the default share option for applications opening this queue for input.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** By default, the open request is for exclusive use of the queue for input.

***YES** By default, the open request is for shared use of the queue for input.

Message delivery sequence (MSGDLYSEQ)

Specifies the message delivery sequence.

The possible values are:

***SAME**
The attribute is unchanged.

***PTY** Messages are delivered in first-in-first-out (FIFO) order within priority.

***FIFO** Messages are delivered in FIFO order regardless of priority.

Harden backout count (HDNBKTCNT)

Specifies whether the count of backed out messages is saved (hardened) across restarts of the message queue manager.

Note: On WebSphere MQ for i5/OS the count is ALWAYS hardened, regardless of the setting of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** The backout count is not hardened.

***YES** The backout count is hardened.

Trigger type (TRGTYPE)

Specifies the condition that initiates a trigger event. When the condition is true, a trigger message is sent to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***FIRST**
When the number of messages on the queue goes from 0 to 1.

***ALL** Every time a message arrives on the queue.

***DEPTH**
When the number of messages on the queue equals the value of the TRGDEPTH attribute.

***NONE**
No trigger messages are written.

Trigger depth (TRGDEPTH)

Specifies, for TRIGTYPE(*DEPTH), the number of messages that initiate a trigger message to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

depth-value
Specify a value ranging from 1 through 999999999.

Trigger message priority (TRGMSGPTY)

Specifies the minimum priority that a message must have before it can produce, or be counted for, a trigger event.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

priority-value
Specify a value ranging from 0 through 9, where 9 is the highest priority.

Trigger data (TRGDATA)

Specifies up to 64 characters of user data that the queue manager includes in the trigger message. This data is made available to the monitoring application that processes the initiation queue, and to the application started by the monitor.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No trigger data is specified.

trigger-data
Specify up to 64 characters enclosed in apostrophes. For a transmission queue you can use this parameter to specify the name of the channel to be started.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Retention interval (RTNITV)

Specifies the retention interval. This interval is the number of hours for which the queue might be needed, based on the date and time when the queue was created.

This information is available to a housekeeping application or an operator and can be used to determine when a queue is no longer required.

Note: The message queue manager does not delete queues, nor does it prevent your queues from being deleted if their retention interval has not expired. It is your responsibility to take any required action.

The possible values are:

***SAME**
The attribute is unchanged.

interval-value
Specify a value ranging from 0 through 999999999.

Maximum queue depth (MAXDEPTH)

Specifies the maximum number of messages allowed on the queue. However, other factors can cause the queue to be treated as full; for example, it appears to be full if there is no storage available for a message.

Note: If this value is subsequently reduced by using the CHGMQMQ command, any messages that are on the queue remain intact even if they cause the new maximum to be exceeded.

The possible values are:

***SAME**
The attribute is unchanged.

depth-value
Specify a value ranging from 0 through 999999999.

Maximum message length (MAXMSGLEN)

Specifies the maximum length for messages on the queue.

Note: If this value is subsequently reduced by using the CHGMQMQ command, any messages that are on the queue remain intact even if they exceed the new maximum length.

Applications might use the value of this attribute to determine the size of buffer they need to retrieve messages from the queue. Therefore change the value only if you know this will not cause an application to operate incorrectly.

The possible values are:

***SAME**
The attribute is unchanged.

length-value
Specify a value ranging from 0 through 100 MB in bytes. The default is 4MB.

Backout threshold (BKTTHLD)

Specifies the backout threshold. WebSphere MQ for i5/OS takes no action based on the value of this attribute, excpet allowing for this attribute to be queried.

The possible values are:

***SAME**
The attribute is unchanged.

threshold-value
Specify a value ranging from 0 through 999999999.

Backout requeue name (BKTQNAME)

Specifies the backout-queue name. WebSphere MQ for i5/OS takes no action based on the value of this attribute, except allowing for this attribute to be queried.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No backout queue is specified.

backout-queue-name
Specify the backout queue name.

Initiation queue (INITQNAME)

Specifies the name of the initiation queue.

Note: The initiation queue must be on the same instance of a message queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No initiation queue is specified.

initiation-queue-name
Specify the initiation queue name.

Usage (USAGE)

Specifies whether the queue is for normal usage, or for transmitting messages to a remote message queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

***NORMAL**
Normal usage (the queue is not a transmission queue)

***TMQ** The queue is a transmission queue that is used to hold messages destined for a remote message queue manager. If the queue is intended for use in situations where a transmission queue name is not explicitly specified, the queue name must be the same as the name of the remote message queue manager. For further information, see WebSphere MQ Intercommunication.

Definition type (DFNTYPE)

Specifies the type of dynamic queue definition that is created when an application issues an MQOPEN API call with the name of this model queue specified in the object descriptor.

Note: This parameter only applies to a model queue definition.

The possible values are:

***SAME**
The attribute is unchanged.

***TEMPDYN**
A temporary dynamic queue is created. This value should not be specified with a DEFMSGPST value of *YES.

***PERMDYN**
A permanent dynamic queue is created.

Target queue (TGTQNAME)

Specifies the name of the object for which this queue is an alias.

The object can be a local or remote queue, a topic or a message queue manager.

Note: The target object does not need to exist at this time but it must exist when a process attempts to open the alias queue.

The possible values are:

***SAME**
The attribute is unchanged.

target-object-name
Specify the name of the target object.

Remote queue (RMTQNAME)

Specifies the name of the remote queue. That is, the local name of the remote queue as defined on the queue manager specified by RMTMQMNAME.

If this definition is used for a queue manager alias definition, RMTQNAME must be blank when the open occurs.

If this definition is used for a reply-to alias, this name is the name of the queue that is to be the reply-to queue.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No remote-queue name is specified (that is, the name is blank). This can be used if the definition is a queue manager alias definition.

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

Note: The name is not checked to ensure that it contains only those characters normally allowed for queue names.

Remote Message Queue Manager (RMTMQMNAME)

Specifies the name of the remote queue manager on which the queue RMTQNAME is defined.

If an application opens the local definition of a remote queue, RMTMQMNAME must not be the name of the connected queue manager. If TMQNAME is blank there must be a local queue of this name, which is to be used as the transmission queue.

If this definition is used for a queue manager alias, RMTMQMNAME is the name of the queue manager, which can be the name of the connected queue manager. Otherwise, if TMQNAME is blank, when the queue is opened there must be a local queue of this name, with USAGE(*TMQ) specified, which is to be used as the transmission queue.

If this definition is used for a reply-to alias, this name is the name of the queue manager that is to be the reply-to queue manager.

The possible values are:

***SAME**
The attribute is unchanged.

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

Note: Ensure this name contains only those characters normally allowed for queue manager names.

Transmission queue (TMQNAME)

Specifies the local name of the transmission queue to be used for messages destined for the remote queue, for either a remote queue or for a queue manager alias definition.

If TMQNAME is blank, a queue with the same name as RMTMQMNAME is used as the transmission queue.

This attribute is ignored if the definition is being used as a queue manager alias and RMTMQMNAME is the name of the connected queue manager.

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

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No specific transmission queue name is defined for this remote queue. The value of this attribute is set to all blanks.

transmission-queue-name
Specify the transmission queue name.

Queue depth high threshold (HIGHTHLD)

Specifies the threshold against which the queue depth is compared to generate a queue depth high event.

The possible values are:

***SAME**
The attribute is unchanged.

threshold-value
Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

Queue depth low threshold (LOWTHLD)

Specifies the threshold against which the queue depth is compared to generate a queue depth low event.

The possible values are:

***SAME**
The attribute is unchanged.

threshold-value
Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

Queue full events enabled (FULLEVT)

Specifies whether queue full events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Queue full events are not generated.

***YES** Queue full events are generated.

Queue high events enabled (HIGHEVT)

Specifies whether queue depth high events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Queue depth high events are not generated.

***YES** Queue depth high events are generated.

Queue low events enabled (LOWEVT)

Specifies whether queue depth low events are generated.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** Queue depth low events are not generated.

***YES** Queue depth low events are generated.

Service interval (SRVITV)

Specifies the service interval. This interval is used for comparison to generate service interval high and service interval OK events.

The possible values are:

***SAME**
The attribute is unchanged.

interval-value
Specify a value ranging from 0 through 999999999. The value is in units of milliseconds.

Service interval events (SRVEVT)

Specifies whether service interval high or service interval OK events are generated.

A service interval high event is generated when a check indicates that no messages have been retrieved from the queue for the time indicated by the SRVITV parameter as a minimum.

A service interval OK event is generated when a check indicates that messages have been retrieved from the queue within the time indicated by the SRVITV parameter.

The possible values are:

***SAME**
The attribute is unchanged.

***HIGH**
Service interval high events are generated.

***OK** Service interval OK events are generated.

***NONE**
No service interval events are generated.

Distribution list support (DISTLIST)

Specifies whether the queue supports distribution lists.

The possible values are:

***SAME**
The attribute is unchanged.

***NO** The queue will not support distribution lists.

***YES** The queue will support distribution lists.

Cluster Name (CLUSTER)

The name of the cluster to which the queue belongs.

Changes to this parameter do not affect instances of the queue that are already open.

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

The possible values are:

***SAME**
The attribute is unchanged.

cluster-name
Only one of the resultant values of CLUSTER or CLUSNL can be non-blank; you cannot specify a value for both.

Cluster Name List (CLUSNL)

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

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

The possible values are:

***SAME**
The attribute is unchanged.

namelist-name
Only one of the resultant values of CLUSTER or CLUSNL can be non-blank; you cannot specify a value for both.

Default Binding (DEFBIND)

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

The possible values are:

***SAME**
The attribute is unchanged.

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

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

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

Cluster Workload Rank (CLWLRANK)

Specifies the cluster workload rank of the queue.

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-rank
Specify a value ranging from 0 through 9.

Cluster Workload Priority (CLWLPRTY)

Specifies the cluster workload priority of the queue.

The possible values are:

***SAME**
The attribute is unchanged.

cluster-workload-priority
Specify a value ranging from 0 through 9.

Cluster workload queue use (CLWLUSEQ)

Specifies the behaviour of an MQPUT when the target queue has both a local instance and at least one remote cluster instance. If the put originates from a cluster channel then this attribute does not apply.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
The value is inherited from the Queue Manager CLWLUSEQ attribute.

***LOCAL**
The local queue will be the sole target of the MQPUT.

***ANY** The queue manager will treat such a local queue as another instance of the cluster queue for the purposes of workload distribution.

Queue Monitoring (MONQ)

Controls the collection of Online Monitoring Data.

Online Monitoring Data is not collected when the queue manager attribute MONQ is set to *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
The collection of online monitoring data is inherited from the setting of the queue manager attribute MONQ.

***OFF** Online monitoring data collection for this queue is switched off.

***LOW** Monitoring data collection is turned on with a low ratio of data collection.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection.

Queue Statistics (STATQ)

Controls the collection of statistics data.

Online monitoring data is not collected when the queue manager attribute STATQ is set to *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

***QMGR**
Statistics data collection is based upon the setting of the queue manager attribute STATQ.

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

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

Queue Accounting (ACCTQ)

Controls the collection of accouting data.

Accounting data is not collected when the queue manager attribute ACCTQ is set to *NONE.

The possible values are:

***SAME**
The attribute is unchanged.

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

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

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

Non Persistent Message Class (NPMCLASS)

Specifies the level of reliability for non-persistent messages put to this queue.

The possible values are:

***SAME**
The attribute is unchanged.

***NORMAL**
Non-persistent messages put to this queue are only lost following a failure,

or a queue manager shutdown. Non-persistent message put to this queue will be discarded in the event of a queue manager restart.

***HIGH**
Non-persistent messages put to this queue are not discarded in the event of a queue manager restart. Non-persistent messages put to this queue may still be lost in the event of a failure.

Message Read Ahead (MSGREADAHD)

Specifies whether non persistent messages are sent to the client ahead of an application requesting them.

The possible values are:

***SAME**
The attribute is unchanged.

***DISABLED**
Read ahead is disabled for this queue. Messages are not sent to the client ahead of an application requesting them regardless of whether read ahead is requested by the client application.

***NO** Non-persistent messages are not sent to the client ahead of an application requesting them. A maximum of one non-persistent message can be lost if the client ends abnormally.

***YES** Non-persistent messages are sent to the client ahead of an application requesting them. Non-persistent messages can be lost if the client ends abnormally or if the client application does not consume all the messages it is sent.

Default Put Response (DFTPUTRESP)

The default put response type (DFTPUTRESP) attribute specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO_RESPONSE_AS_Q_DEF option.

The possible values are:

***SAME**
The attribute is unchanged.

***SYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are issued as if MQPMO_SYNC_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application. This is the default value supplied with WebSphere MQ, but your installation might have changed it.

***ASYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are always issued as if MQPMO_ASYNC_RESPONSE had been specified instead. Some fields in

the MQMD and MQPMO are not returned by the queue manager to the application; but an improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

Property Control (PROPCTL)

Specifies what happens to properties of messages that are retrieved from queues using the MQGET call when the MQGMO_PROPERTIES_AS_Q_DEF option is specified.

The possible values are:

***SAME**
The attribute is unchanged.

***COMPAT**
If the message contains a property with a prefix of ″mcd.″, ″jms.″, ″usr.″ or ″mqext.″ then all message properties are delivered to the application in an MQRFH2 header. Otherwise all properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

***NONE**
All properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

***ALL** All properties of the message, except those contained in the message descriptor (or extension), are contained in one or more MQRFH2 headers in the message data.

***FORCE**
Properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle.

Target Type (TARGTYPE)

Specifies the type of object to which the alias resolves.

The possible values are:

***SAME**
The attribute is unchanged.

***QUEUE**
Queue object.

***TOPIC**
Topic object.

Examples

None

Error messages

Unknown

Copy MQ Subscription (CPYMQMSUB)

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

The Copy MQ Subscription (CPYMQMSUB) command creates an MQ subscription of the same type and, for attributes not specified in the command, with the same attribute values as an existing subscription.

Parameters

Keyword	Description	Choices	Notes
FROMSUB	From subscription	*Character value*, ***SAME**	Optional, Key, Positional 1
FROMSUBID	From subscription identifier	*Character value*, ***SAME**	Optional, Key, Positional 2
TOSUB	To subscription	*Character value*	Optional, Key, Positional 3
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 4
REPLACE	Replace	***NO**, *YES	Optional, Positional 5
TOPICSTR	Topic string	*Character value*, *NONE, ***SAME**	Optional, Positional 6
TOPICOBJ	Topic object	*Character value*, *NONE, ***SAME**	Optional, Positional 7
DEST	Destination	*Character value*, ***SAME**	Optional, Positional 8
DESTMQM	Destination Queue Manager	*Character value*, *NONE, ***SAME**	Optional, Positional 9
DESTCRRLID	Destination Correlation Id	*Character value*, *NONE, ***SAME**	Optional, Positional 10
PUBACCT	Publish Accounting Token	*Character value*, *NONE, ***SAME**	Optional, Positional 11
PUBAPPID	Publish Application Id	*Character value*, *NONE, ***SAME**	Optional, Positional 12
SUBUSER	Subscription User Id	*Character value*, *NONE, ***SAME**	Optional, Positional 13

Keyword	Description	Choices	Notes
USERDATA	Subscription User Data	*Character value*, *NONE, ***SAME**	Optional, Positional 14
SELECTOR	Selector String	*Character value*, *NONE, ***SAME**	Optional, Positional 15
PSPROP	PubSub Property	***SAME**, *NONE, *COMPAT, *RFH2	Optional, Positional 16
DESTCLASS	Destination Class	***SAME**, *MANAGED, *PROVIDED	Optional, Positional 17
SUBSCOPE	Subscription Scope	***SAME**, *ALL, *QMGR	Optional, Positional 18
VARUSER	Variable User	***SAME**, *ANY, *FIXED	Optional, Positional 19
REQONLY	Request Publications	***SAME**, *YES, *NO	Optional, Positional 20
PUBPTY	Publish Priority	0-9, ***SAME**, *ASPUB, *ASQDEF	Optional, Positional 21
WSCHEMA	Wildcard Schema	***SAME**, *CHAR, *TOPIC	Optional, Positional 22
EXPIRY	Expiry Time	0-999999999, ***SAME**, *UNLIMITED	Optional, Positional 23

From subscription (FROMSUB)

Specifies the name of the existing subscription to provide values for the attributes not specified in this command.

The possible values are:

from-subscription-name
Specify a maximum of 256 bytes for the subscription name.

Note: Subscription names of greater than 256 bytes can be speicifed using MQSC.

From subscription identifier (FROMSUBID)

Specifies the subscription identifier of the existing subscription to provide values for the attributes not specified in this command.

The possible values are:

from-subscription-identifier
Specify the 48 character hexadecimal string representing the 24 byte subscription identifier.

To subscription (TOSUB)

The name of the new subscription to be created.

Note: Subscription names of greater than 256 bytes can be speicifed using MQSC.

If a subscription with this name already exists, REPLACE(*YES) must be specified.

The possible values are:

to-subscription-name
Specify a maximum of 256 bytes for name of the MQ subscription being created.

Note: Subscription names of greater than 256 bytes can be speicifed using MQSC.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Replace (REPLACE)

Specifies whether the new subscription should replace an existing subscription with the same name.

The possible values are:

***NO** This subscription does not replace any existing subscription with the same name or subscription identifier. The command fails if the subscription already exists.

***YES** Replace the existing subscription. If there is no subscription with the same name or subscription identifier, a new subscription is created.

Topic string (TOPICSTR)

Specifies the topic string associated with this subscription.

The possible values are:

topic-string
Specify a maximum of 256 bytes for the topic string.

Note: Topic strings of greater than 256 bytes can be speicifed using MQSC.

Topic object (TOPICOBJ)

Specifies the topic object associated with this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

topic-object
Specify the name of the topic object.

Destination (DEST)

Specifies the destination queue for messages published to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

destination-queue
Specify the name of the destination queue.

Destination Queue Manager (DESTMQM)

Specifies the destination queue manager for messages published to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No destination queue manager is specified.

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

Destination Correlation Id (DESTCRRLID)

Specifies the correlation identifier for messages published to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Messages are placed on the destination with a correlation identifier of MQCI_NONE.

correlation-identifier
Specify the 48 character hexadecimal string representing the 24 byte correlation identifier.

Publish Accounting Token (PUBACCT)

Specifies the accounting token for messages published to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Messages are placed on the destination with an accounting token of MQACT_NONE.

publish-accounting-token
Specify the 64 character hexadecimal string representing the 32 byte publish accounting token.

Publish Application Id (PUBAPPID)

Specifies the publish application identity for messages published to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No publish application identifier is specified.

publish-application-identifier
Specify the publish application identifier.

Subscription User Id (SUBUSER)

Specifies the user profile that 'owns' this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No user profile is specified.

user-profile
Specify the user profile.

Subscription User Data (USERDATA)

Specifies the user data associated with the subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No user data is specified.

user-data
Specify a maximum of 256 bytes for user data.

Note: User data of greater than 256 bytes can be speicifed using MQSC.

Selector String (SELECTOR)

Specifies the SQL 92 selector string to be applied to messages published on the named topic to select whether they are eligible for this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
No selection string is specified.

selection-string
Specify a maximum of 256 bytes for selection string.

Note: Selection strings of greater than 256 bytes can be speicifed using MQSC.

PubSub Property (PSPROP)

Specifies the manner in which publish / subscribe related message properties are added to messages sent to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***NONE**
Publish / subscribe properties are not added to the message.

***COMPAT**
Publish / subscribe properties are added to the message to maintain compatibility with V6 Publish / Subscribe.

***RFH2**
Publish / subscribe properties are added to the message within an RFH Version 2 header.

Destination Class (DESTCLASS)

Specifies whether this is a managed subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***MANAGED**
The destination is managed.

***PROVIDED**
The destination is a queue.

Subscription Scope (SUBSCOPE)

Specifies whether this subscription should be forwarded (as a proxy subscription) to other brokers, so that the subscriber will receive messages published at those other brokers.

The possible values are:

***SAME**
The attribute is unchanged.

***ALL** The subscription will be forwarded to all queue managers directly connected via a publish / subscribe collective or hierarchy.

***QMGR**
The subscription will only forward messages published on the topic within this queue manager.

Variable User (VARUSER)

Specifies whether user profiles other than the creator of the subscription can connect to it (subject to topic and destination authority checks).

The possible values are:

***SAME**
The attribute is unchanged.

***ANY** Any user profiles can connect to the subsciption.

***FIXED**
Only the user profile that created the subscription can connect to it.

Request Publications (REQONLY)

Specifies whether the subscriber will poll for updates via MQSUBRQ API, or whether all publications are delivered to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***YES** Publications are only delivered to this subscription in response to an MQSUBRQ API.

***NO** All publications on the topic are delivered to this subscription.

Publish Priority (PUBPTY)

Specifies the priority of the message sent to this subscription.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPUB**
The priority of the message sent to this subscription is taken from that supplied in the published message.

***ASQDEF**
The priority of the message sent to this subscription is taken from the default priority of the queue defined as the destination.

priority-value
Specify a priority ranging from 0 through 9.

Wildcard Schema (WSCHEMA)

Specifies the schema to be used when interpreting wild card characters in the topic string.

The possible values are:

***SAME**
The attribute is unchanged.

***TOPIC**
Wildcard characters represent portions of the topic hierarchy.

***CHAR**
Wildcard characters represent portions of strings.

Expiry Time (EXPIRY)

Specifies the expiry time of the subscription. After a subscription's expiry time has elapsed, it becomes eligible to be discarded by the queue manager and will receive no further publications.

The possible values are:

***SAME**
The attribute is unchanged.

***UNLIMITED**
The subscription does not expire.

expiry-time
Specify an expiry time in tenths of a second ranging from 0 through 999999999.

Examples

None

Error messages

Unknown

Copy MQ Service (CPYMQMSVC)

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

The Copy MQ Service (CPYMQMSVC) command creates an MQ service definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing service definition.

Parameters

Keyword	Description	Choices	Notes
FROMSVC	From Service	*Character value*	Required, Key, Positional 1
TOSVC	To Service	*Character value*	Required, Key, Positional 2

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO**, *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SAME**	Optional, Positional 5
STRCMD	Start program	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 6
	Qualifier 1: Start program	*Name*	
	Qualifier 2: Library	*Name*	
STRARG	Start program arguments	*Character value*, *BLANK, ***SAME**	Optional, Positional 7
ENDCMD	End program	Single values: ***SAME**, *NONE Other values: *Qualified object name*	Optional, Positional 8
	Qualifier 1: End program	*Name*	
	Qualifier 2: Library	*Name*	
ENDARG	End program arguments	*Character value*, *BLANK, ***SAME**	Optional, Positional 9
STDOUT	Standard output	*Character value*, *BLANK, ***SAME**	Optional, Positional 10
STDERR	Standard error	*Character value*, *BLANK, ***SAME**	Optional, Positional 11
TYPE	Service type	***SAME**, *CMD, *SVR	Optional, Positional 12
CONTROL	Service control	***SAME**, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 13

From Service (FROMSVC)

Specifies the name of the existing service definition to provide values for the attributes not specified in this command.

The possible values are:

from-service-name
Specify the name of the source service.

To Service (TOSVC)

The name of the new service definition to be created. The name can contain a maximum of 48 characters.

If a service definition with this name already exists, REPLACE(*YES) must be specified.

The possible values are:

to-service-name
Specify the name of the service being created.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Replace (REPLACE)

Specifies whether the new service definition should replace an existing service definition with the same name.

The possible values are:

***NO** This definition does not replace any existing service definition with the same name. The command fails if the named service definition already exists.

***YES** Replace the existing service definition. If there is no definition with the same name, a new definition is created.

Text 'description' (TEXT)

Specifies text that briefly describes the service definition.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Start program (STRCMD)

The name of the program to run.

The possible values are:

***SAME**
The attribute is unchanged.

start-command
The name of the start command executable.

Start program arguments (STRARG)

The arguments passed to the program at startup.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
No arguments are passed to the start command.

start-command-arguments
The arguments passed to the start command.

End program (ENDCMD)

The name of the executable to run when the service is requested to stop.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
No end command is executed.

end-command
The name of the end command executable.

End program arguments (ENDARG)

The arguments passed to the end program when the service is requested to stop.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
No arguments are passed to the end command.

end-command-arguments
The arguments passed to the end command.

Standard output (STDOUT)

The path to a file to which the standard output of the service program is redirected.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The standard output is discarded.

stdout-path
The standard output path.

Standard error (STDERR)

The path to a file to which the standard error of the service program is redirected.

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The standard error is discarded.

stderr-path
The standard error path.

Service type (TYPE)

Mode in which to run service.

The possible values are:

***SAME**
The attribute is unchanged.

***CMD** When started the command is executed but no status is collected or displayed.

***SVR** The status of the executable started will be monitored and displayed.

Service control (CONTROL)

Whether the service should be started automatically at queue manager start.

The possible values are:

***SAME**
The attribute is unchanged.

***MANUAL**
The service is automatically started or stopped.

***QMGR**
The service is started and stopped as the queue manager is started and stopped.

***STARTONLY**
The service is started as the queue manager is started, but will not be requested to stop when the queue manager is stopped.

Examples

None

Error messages

Unknown

Copy MQ Topic (CPYMQMTOP)

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

The Copy MQ Topic (CPYMQMTOP) command creates an MQ topic object of the same type and, for attributes not specified in the command, with the same attribute values as an existing topic object.

Parameters

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

From topic (FROMTOP)

Specifies the name of the existing topic object to provide values for the attributes not specified in this command.

The possible values are:

from-topic-name
Specify the name of the source MQ topic.

To topic (TOTOP)

The name of the new topic object to be created. The name can contain a maximum of 48 characters.

If a topic object with this name already exists, REPLACE(*YES) must be specified.

The possible values are:

to-topic-name
Specify the name of the MQ topic being created.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Replace (REPLACE)

Specifies whether the new topic object should replace an existing topic object with the same name.

The possible values are:

***NO** This object does not replace any existing topic object with the same name. The command fails if the named topic object already exists.

***YES** Replace the existing topic object. If there is no object with the same name, a new object is created.

Text 'description' (TEXT)

Specifies text that briefly describes the topic object.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SAME**
The attribute is unchanged.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Topic string (TOPICSTR)

Specifies the topic string represented by this topic object definition.

The possible values are:

topic-string
Specify a maximum of 256 bytes for the topic string.

Note: Topic strings of greater than 256 bytes can be speicifed using MQSC.

Durable subscriptions (DURSUB)

Specifies whether applications are permitted to make durable subscriptions on this topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
Whether durable subscriptions can be made on this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***YES** Durable subscriptions can be made on this topic.

***NO** Durable subscriptions cannot be made on this topic.

Durable model queue (MGDDURMDL)

Specifies the name of the model queue to be used for durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

***SAME**
The attribute is unchanged.

durable-model-queue
Specify the name of the model queue.

Non-durable model queue (MGDNDURMDL)

Specifies the name of the model queue to be used for non-durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

***SAME**
The attribute is unchanged.

non-durable-model-queue
Specify the name of the model queue.

Publish (PUBENBL)

Specifies whether messages can be published to the topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
Whether messages can be published to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***YES** Messages can be published to the topic.

***NO** Messages cannot be published to the topic.

Subscribe (SUBENBL)

Specifies whether applications are to be permitted to subscribe to this topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
Whether applications can subscribe to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***YES** Subscriptions can be made to this topic.

***NO** Applications cannot subscribe to this topic.

Default message priority (DFTPTY)

Specifies the default priority of messages published to the topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
The default priority is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

priority-value
Specify a value ranging from 0 through 9.

Default message persistence (DFTMSGPST)

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

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
The default persistence is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

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

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

Default Put Response (DFTPUTRESP)

Specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO_RESPONSE_AS_Q_DEF option.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
The default response type is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***SYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are issued as if MQPMO_SYNC_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application.

***ASYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are always issued as if

MQPMO_ASYNC_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application. An improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

Wildcard behaviour (WILDCARD)

Specifies the behaviour of wildcard subscriptions with respect to this topic.

The possible values are:

***SAME**
The attribute is unchanged.

***PASSTHRU**
Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will receive publications made to this topic and to topic strings more specific than this topic.

***BLOCK**
Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will not receive publications made to this topic or to topic strings more specific than this topic.

Persistent message delivery (PMSGDLV)

Specifies the delivery mechanism for persistent messages published to this topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***ALL** Persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

***ALLDUR**
Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

***ALLAVAIL**
Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

Non-persistent message delivery (NPMSGDLV)

Specifies the delivery mechanism for non-persistent messages published to this topic.

The possible values are:

***SAME**
The attribute is unchanged.

***ASPARENT**
The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***ALL** Non-persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

***ALLDUR**
Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

***ALLAVAIL**
Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

Examples

None

Error messages

Unknown

Create Message Queue Manager (CRTMQM)

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

The Create Message Queue Manager (CRTMQM) command creates a local queue manager that can be started with the Start Message Queue Manager (STRMQM) command.

Parameters

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

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

queue-manager-name

Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Text 'description' (TEXT)

Specifies text that briefly describes the queue manager definition.

The possible values are:

***BLANK**
No text is specified.

description
Specify no more than 64 characters enclosed in apostrophes.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Trigger interval (TRGITV)

Specifies the trigger time interval, expressed in milliseconds, for use with queues that have TRGTYPE(*FIRST) specified.

When the arrival of a message on a queue causes a trigger message to be put on the initiation queue, then any message that arrives on the same queue within the specified interval does not cause another trigger message to be put on the initiation queue.

The possible values are:

999999999
The trigger time interval is 999999999 milliseconds.

interval-value
Specify a value in milliseconds, in the range 0 through 999999999.

Undelivered message queue (UDLMSGQ)

Specifies the name of the local queue that is to be used for undelivered messages. Messages are put on this queue if they cannot be routed to their correct destination.

The possible values are:

***NONE**
There is no undelivered-message queue. The attribute is set to a blank string.

undelivered-message-queue-name
Specify the name of a local queue that is to be used as the undelivered-message queue.

Default transmission queue (DFTTMQ)

Specifies the name of the local transmission queue that is to be used as the default transmission queue. Messages transmitted to a remote queue manager are put on the default transmission queue if there is no transmission queue defined for their destination.

The possible values are:

***NONE**
There is no default transmission queue. The attribute is set to a blank string.

default-transmission-queue-name
Specify the name of a local transmission queue that is to be used as the default transmission queue.

Maximum handle limit (MAXHDL)

Specifies the maximum number of handles that any one job can have open at the same time.

The possible values are:

256 The default number of open handles is 256.

maximum-handle-limit
Specify a value in the range 0 through 999999999.

Maximum uncommitted messages (MAXUMSG)

Specifies the maximum number of uncommitted messages. That is:

- The number of messages that can be retrieved, plus
- The number of messages that can be put on a queue, plus
- Any trigger messages generated within this unit of work,

under any one syncpoint. This limit does not apply to messages that are retrieved or put outside syncpoint.

The possible values are:

10000 The default value is 10000 uncommitted messages.

maximum-uncommitted-messages
Specify a value in the range 1 through 999999999.

Default Queue Manager (DFTQMGR)

Specifies whether the queue manager being created is the default queue manager.

The possible values are:

***NO** The queue manager is not to be the default queue manager.

***YES** The queue manager is to be the default queue manager.

Queue Manager Library (MQMLIB)

Specifies the library to be used by the queue manager.

The possible values are:

***AUTO**
The library to be used by the queue manager is chosen automatically.

library name
Specify the library to be used by the queue manager.

ASP Number (ASP)

Specifies the auxillary storage pool from which the system allocates storage for the queue manager library, journal and journal receivers.

The possible values are:

***SYSTEM**
The system auxillary storage pool (ASP 1) provides the storage for the queue manager library, journal and journal receivers.

auxillary-storage-pool-number
Specify a value in the range 1 through 32 to specify the number of the system or basic user ASP to provide storage for the queue manager library, journal and journal receivers.

ASPs allow isolation of objects on one or more specific disk units. This may reduce the loss of data due to a disk media failure. In most cases, only data that is stored on disk units in the affected ASP is lost. It is recommended to store the queue manager library and journal data in a seperate user ASP to that of the root IFS file system, to provide failover and reduce disk contention.

Journal receiver threshold (THRESHOLD)

Specifies the threshold in kilobytes for the queue managers journal receivers.

The possible values are:

threshold-value
Specify a value in the range 100000 through 1000000000 in kilobytes (KB) of storage. Each 1000 KB specifies 1024000 bytes of storage space. When the size of the space for the journal receiver is larger than the size specified by this value, a message is sent to the identified message queue if appropriate, and journaling continues.

Journal buffer size (JRNBUFSIZ)

Specifies the journal buffer size in bytes

The possible values are:

journal-buffer-size
Specify a value in bytes, in the range 32000 through 15761440.

Examples

None

Error messages

Unknown

Create MQ AuthInfo object (CRTMQMAUTI)

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

The Create MQ AuthInfo object (CRTMQMAUTI) command creates a new authentication information object, specifying those attributes that are different from the system default.

Parameters

Keyword	Description	Choices	Notes
AINAME	AuthInfo name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, *DFT	Required, Key, Positional 2
AUTHTYPE	AuthInfo type	*CRLLDAP	Required, Key, Positional 3
CONNAME	Connection name	*Character value*, *SYSDFTAI	Required, Key, Positional 4
REPLACE	Replace	***NO**, *YES	Optional, Positional 5
TEXT	Text 'description'	*Character value*, ***SYSDFTAI**, *NONE	Optional, Positional 6
USERNAME	User name	*Character value*, ***SYSDFTAI**, *NONE	Optional, Positional 7
PASSWORD	User password	*Character value*, ***SYSDFTAI**, *NONE	Optional, Positional 8

AuthInfo name (AINAME)

The name of the new authentication information object to create.

The possible values are:

authentication-information-name
Specify the name of the authentication information object. The maximum string length is 48 characters.

Message Queue Manager name (MQMNAME)

The name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of an existing message queue manager. The maximum string length is 48 characters.

AuthInfo type (AUTHTYPE)

The type of the authentication information object. There is no default value

The possible values are:

***CRLLDAP**
The type of the authentication information object is CRLLDAP.

Connection name (CONNAME)

The DNS name or IP address of the host on which the LDAP server is running, together with an optional port number. The default port number is 389. No default is provided for the DNS name or IP address.

The possible values are:

***SYSDFTAI**
The connection name is set to the system default value in SYSTEM.DEFAULT.AUTHINFO.CRLLDAP.

connection-name
Specify the fully qualified DNS name or IP address of the host together with an optional port number. The maximum string length is 264 characters.

Replace (REPLACE)

If an authentication information object with the same name already exists, this specifies whether it is replaced.

The possible values are:

***NO** This definition does not replace any existing authentication information object with the same name. The command fails if the named authentication information object already exists.

***YES** Replace an existing authentication information object. A new object is created if the named authentication information object does not exist.

Text 'description' (TEXT)

A short text description of the authentication information object.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SYSDFTAI**
The text string is set to the system default value in SYSTEM.DEFAULT.AUTHINFO.CRLLDAP.

***NONE**
The text is set to a blank string.

description
The string length can be up to 64 characters enclosed in apostrophes.

User name (USERNAME)

The distinguished name of the user that is binding to the directory. The default user name is blank.

The possible values are:

***SYSDFTAI**
The user name is set to the system default value in SYSTEM.DEFAULT.AUTHINFO.CRLLDAP.

***NONE**
The user name is blank.

LDAP-user-name
Specify the Distinguished name of the LDAP user. The maximum string length is 1024 characters.

User password (PASSWORD)

The password for the LDAP user.

The possible values are:

***SYSDFTAI**
The password is set to the system default value in SYSTEM.DEFAULT.AUTHINFO.CRLLDAP.

***NONE**
The password is blank.

LDAP-password
The LDAP user password. The maximum string length is 32 characters.

Examples

None

Error messages

Unknown

Create MQ Channel (CRTMQMCHL)

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

The Create MQ Channel (CRTMQMCHL) command creates a new MQ channel definition, specifying those attributes that are to be different from the default values.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*	Required, Key, Positional 1
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN	Required, Key, Positional 2

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO** , *YES	Optional, Positional 4
TRPTYPE	Transport type	*LU62, *TCP, ***SYSDFTCHL**	Optional, Positional 5
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTCHL**	Optional, Positional 6
TGTMQMNAME	Target Queue Manager	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 7
CONNAME	Connection name	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 8
TPNAME	Transaction Program Name	*Character value*, *BLANK, ***SYSDFTCHL**	Optional, Positional 9
MODENAME	Mode Name	*Character value*, *BLANK, ***SYSDFTCHL**	Optional, Positional 10
TMQNAME	Transmission queue	*Character value*, ***SYSDFTCHL**	Optional, Positional 11
MCANAME	Message channel agent	Single values: ***SYSDFTCHL** , *NONE Other values: *Qualified object name*	Optional, Positional 12
	Qualifier 1: Message channel agent	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MCAUSRID	Message channel agent user ID	*Character value*, *NONE, *PUBLIC, ***SYSDFTCHL**	Optional, Positional 13
MCATYPE	Message channel agent Type	*PROCESS, *THREAD, ***SYSDFTCHL**	Optional, Positional 14
BATCHINT	Batch Interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 15
BATCHSIZE	Batch size	1-9999, ***SYSDFTCHL**	Optional, Positional 16
DSCITV	Disconnect interval	0-999999, ***SYSDFTCHL**	Optional, Positional 17

Keyword	Description	Choices	Notes
SHORTTMR	Short retry interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 18
SHORTRTY	Short retry count	0-999999999, ***SYSDFTCHL**	Optional, Positional 19
LONGTMR	Long retry interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 20
LONGRTY	Long retry count	0-999999999, ***SYSDFTCHL**	Optional, Positional 21
SCYEXIT	Security exit	Single values: ***SYSDFTCHL** , *NONE Other values: *Qualified object name*	Optional, Positional 22
	Qualifier 1: Security exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CSCYEXIT	Security exit	*Character value*, ***SYSDFTCHL** , *NONE	Optional, Positional 23
SCYUSRDATA	Security exit user data	*Character value*, ***SYSDFTCHL** , *NONE	Optional, Positional 24
SNDEXIT	Send exit	Single values: ***SYSDFTCHL** , *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 25
	Qualifier 1: Send exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CSNDEXIT	Send exit	Single values: ***SYSDFTCHL** , *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 26
SNDUSRDATA	Send exit user data	Values (up to 10 repetitions): *Character value*, ***SYSDFTCHL** , *NONE	Optional, Positional 27
RCVEXIT	Receive exit	Single values: ***SYSDFTCHL** , *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 28
	Qualifier 1: Receive exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
CRCVEXIT	Receive exit	Single values: ***SYSDFTCHL** , *NONE Other values (up to 10 repetitions): *Character value*	Optional, Positional 29
RCVUSRDATA	Receive exit user data	Values (up to 10 repetitions): *Character value*, ***SYSDFTCHL** , *NONE	Optional, Positional 30

Keyword	Description	Choices	Notes
MSGEXIT	Message exit	Single values: ***SYSDFTCHL** , *NONE Other values (up to 10 repetitions): *Qualified object name*	Optional, Positional 31
	Qualifier 1: Message exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGUSRDATA	Message exit user data	Values (up to 10 repetitions): *Character value*, ***SYSDFTCHL** , *NONE	Optional, Positional 32
MSGRTYEXIT	Message retry exit	Single values: ***SYSDFTCHL** , *NONE Other values: *Qualified object name*	Optional, Positional 33
	Qualifier 1: Message retry exit	*Name*	
	Qualifier 2: Library	*Name*, ***CURLIB**	
MSGRTYDATA	Message retry exit data	*Character value*, ***SYSDFTCHL** , *NONE	Optional, Positional 34
MSGRTYNBR	Number of message retries	0-999999999, ***SYSDFTCHL**	Optional, Positional 35
MSGRTYITV	Message retry interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 36
CVTMSG	Convert message	*YES, *NO, ***SYSDFTCHL**	Optional, Positional 37
PUTAUT	Put authority	*DFT, *CTX, ***SYSDFTCHL**	Optional, Positional 38
SEQNUMWRAP	Sequence number wrap	100-999999999, ***SYSDFTCHL**	Optional, Positional 39
MAXMSGLEN	Maximum message length	0-104857600, ***SYSDFTCHL**	Optional, Positional 40
HRTBTINTVL	Heartbeat interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 41
NPMSPEED	Non Persistent Message Speed	*FAST, *NORMAL, ***SYSDFTCHL**	Optional, Positional 42
CLUSTER	Cluster Name	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 43
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 44

Keyword	Description	Choices	Notes
NETPRTY	Network Connection Priority	0-9, ***SYSDFTCHL**	Optional, Positional 45
SSLCIPH	SSL CipherSpec	*Character value*, *NULL_MD5, *NULL_SHA, *RC4_MD5_EXPORT, *RC4_MD5_US, *RC4_SHA_US, *RC2_MD5_EXPORT, *DES_SHA_EXPORT, *TRIPLE_DES_SHA_US, *AES_SHA_US, *TLS_RSA_WITH_NULL_MD5, *TLS_RSA_WITH_NULL_SHA, *TLS_RSA_EXPORT_WITH_RC4_40_MD5, *TLS_RSA_WITH_RC4_128_MD5, *TLS_RSA_WITH_RC4_128_SHA, *TLS_RSA_EXPORT_WITH_RC2_40_MD5, *TLS_RSA_WITH_DES_CBC_SHA, *TLS_RSA_WITH_3DES_EDE_CBC_SHA, *TLS_RSA_WITH_AES_128_CBC_SHA, *TLS_RSA_WITH_AES_256_CBC_SHA, *NONE, ***SYSDFTCHL**	Optional, Positional 46
SSLCAUTH	SSL Client Authentication	*REQUIRED, *OPTIONAL, ***SYSDFTCHL**	Optional, Positional 47
SSLPEER	SSL Peer name	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 48
LOCLADDR	Local communication address	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 49
BATCHHB	Batch Heartbeat Interval	0-999999999, ***SYSDFTCHL**	Optional, Positional 50
USERID	Task user identifier	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 51
PASSWORD	Password	*Character value*, *NONE, ***SYSDFTCHL**	Optional, Positional 52
KAINT	Keep Alive Interval	*Integer*, *AUTO, ***SYSDFTCHL**	Optional, Positional 53
COMPHDR	Header Compression	Values (up to 2 repetitions): *NONE, *SYSTEM, ***SYSDFTCHL**	Optional, Positional 54
COMPMSG	Message Compression	Single values: *ANY Other values (up to 4 repetitions): *NONE, *RLE, *ZLIBHIGH, *ZLIBFAST, ***SYSDFTCHL**	Optional, Positional 55
MONCHL	Channel Monitoring	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SYSDFTCHL**	Optional, Positional 56
STATCHL	Channel Statistics	*QMGR, *OFF, *LOW, *MEDIUM, *HIGH, ***SYSDFTCHL**	Optional, Positional 57

Keyword	Description	Choices	Notes
CLWLRANK	Cluster Workload Rank	0-9, ***SYSDFTCHL**	Optional, Positional 58
CLWLPRTY	Cluster Workload Priority	0-9, ***SYSDFTCHL**	Optional, Positional 59
CLWLWGHT	Cluster Channel Weight	1-99, ***SYSDFTCHL**	Optional, Positional 60
SHARECNV	Sharing Conversations	0-999999999, ***SYSDFTCHL**	Optional, Positional 61
PROPCTL	Property Control	*COMPAT, *NONE, *ALL, ***SYSDFTCHL**	Optional, Positional 62
MAXINST	Maximum Instances	0-999999999, ***SYSDFTCHL**	Optional, Positional 63
MAXINSTC	Maximum Instances Per Client	0-999999999, ***SYSDFTCHL**	Optional, Positional 64
CLNTWGHT	Client Channel Weight	0-99, ***SYSDFTCHL**	Optional, Positional 65
AFFINITY	Client Channel Affinity	*PREFERRED, *NONE, ***SYSDFTCHL**	Optional, Positional 66

Channel name (CHLNAME)

Specifies the name of the new channel definition; the name can contain a maximum of 20 characters. Channel names must be unique. If a channel definition with this name already exists, REPLACE(*YES) must be specified.

Channel type (CHLTYPE)

Specifies the type of the channel being defined.

The possible values are:

***SDR** Sender channel

***SVR** Server channel

***RCVR**
Receiver channel

***RQSTR**
Requester channel

***SVRCN**
Server-connection channel

***CLUSSDR**
Cluster-sender channel

***CLUSRCVR**
Cluster-receiver channel

***CLTCN**
Client-connection channel

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Replace (REPLACE)

Specifies whether the new channel definition should replace an existing channel definition with the same name.

The possible values are:

***NO** Do not replace the existing channel definition. The command fails if the named channel definition already exists.

***YES** Replace the existing channel definition. If there is no definition with the same name, a new definition is created.

Transport type (TRPTYPE)

Specifies the transmission protocol.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***LU62** SNA LU 6.2.

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

Text 'description' (TEXT)

Specifies text that briefly describes the channel definition.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Target Queue Manager (TGTMQMNAME)

Specifies the name of the target queue manager.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The name of the target queue manager for a client connection channel (CHLTYPE) *CLTCN is unspecified.

message-queue-manager-name
The name of the target message queue manager for a client connection channel (CHLTYPE) *CLTCN.

For other channel types this parameter must not be specified.

Connection name (CONNAME)

Specifies the name of the machine to connect.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The connection name is blank.

connection-name
Specify the connection name as required by the transmission protocol:

- For *LU62, specify the name of the CSI object.

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

  If a connection name is not specified for cluster-receiver channels a connection name is automatically generated, assuming the default port and using the current IP address of the system.

  Where a port is not specified the default port 1414 is assumed.

For cluster-receiver channels the connection name relates to the local queue manager, and for other channels it relates to the target queue manager.

This parameter is required for channels with channel type (CHLTYPE) of *SDR, *RQSTR, *CLTCN and *CLUSSDR. It is optional for *SVR and *CLUSRCVR channels, and is not valid for *RCVR or *SVRCN channels.

Transaction Program Name (TPNAME)

This parameter is valid for channels with a TRPTYPE defined as LU 6.2 only.

This parameter must be set to the SNA transaction program name, unless the CONNAME contains a side-object name in which case it must be set to blanks. The name is taken instead from the CPI-C Communications Side Object.

This parameter is not valid for channels with a CHLTYPE defined as *RCVR.

The possible values are:

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

***NONE**
No transaction program name is specified.

***BLANK**
The transaction program name is taken from CPI-C Communications Side Object. The side object name must be specified in the CONNAME parameter.

Transaction Program Name
Specify the SNA transaction program name.

Mode Name (MODENAME)

This parameter is valid for channels with a TRPTYPE defined as LU 6.2. If TRPTYPE is not defined as LU 6.2 the data is ignored and no error message is issued.

If specified, the value must be set to the SNA mode name, unless the CONNAME contains a side-object name, in which case it must be set to blanks. The name is then taken from the CPI-C Communications Side Object.

This parameter is not valid for channels with CHLTYPE defined as *RCVR or *SVRCONN.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***BLANK**
Name will be taken from the CPI-C Communications Side Object. This must be specified in the CONNAME parameter.

***NONE**
No mode name is specified.

SNA-mode-name
Specify the SNA Mode Name

Transmission queue (TMQNAME)

Specifies the name of the transmission queue.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

transmission-queue-name
Specify the name of the transmission queue.

A transmission queue name is required if the channel type (CHLTYPE) is *SDR or *SVR. For other channel types, the parameter must not be specified.

Message channel agent (MCANAME)

This parameter is reserved and should not be used.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The MCA program name is blank.

This parameter cannot be specified for a channel type (CHLTYPE) of *RCVR, *SVRCN, or *CLTCN.

Message channel agent user ID (MCAUSRID)

Specifies the message channel agent user identifier which is to be used by the message channel agent for authorization to access MQ resources, including (if PUTAUT is *DFT) authorization to put the message to the destination queue for receiver or requester channels.

The possible values are:

***SYSDFTCHL**
The value is taken from the system default channel for the type of the channel being created.

***NONE**
The message channel agent uses its default user identifier.

***PUBLIC**
Uses the public authority.

mca-user-identifier
Specify the user identifier to be used.

This parameter cannot be specified for a channel type (CHLTYPE) of *CLTCN.

Message channel agent Type (MCATYPE)

Specifies whether the message-channel-agent program should run as a thread or a process.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***PROCESS**
The message channel agent runs as a separate process.

***THREAD**
The message channel agent runs as a separate thread.

This parameter can only be specified for a channel type (CHLTYPE) of *SDR, *SVR, *RQSTR, *CLUSSDR or *CLUSRCVR.

Batch Interval (BATCHINT)

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

The batch is terminated by which ever of the following occurs first: BATCHSZ messages have been sent, or the transmission queue is empty and BATCHINT is exceeded.

The default value is 0, which means that the batch is terminated as soon as the transmission queue becomes empty (or the BATCHSZ limit is reached).

The value must be in the range 0 through 999999999.

This parameter is valid for channels with CHLTYPE defined as *SDR, *SVR, *CLUSSDR, or *CLUSRCVR.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

batch-interval
Specify a value in the range 0 through 999999999. A value of 0 indicates the batch will be terminated as soon as the transmission queue is empty,

Batch size (BATCHSIZE)

Specifies the maximum number of messages that should be sent down a channel before a checkpoint is taken.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

batch-size
Specify a value in the range 1 through 9999

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Disconnect interval (DSCITV)

Specifies the disconnect interval, which defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before closing the channel.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

disconnect-interval
Specify a value in the range 0 through 999999. A value of 0 indicates an indefinite wait.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR or *CLTCN.

Short retry interval (SHORTTMR)

Specifies the short retry wait interval for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. This defines the interval between attempts to establish a connection to the remote machine.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

short-retry-interval
Specify a value in the range 0 through 999999999.

Note: For implementation reasons, the maximum retry interval that can be used is 999999; values exceeding this are treated as 999999.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR, *CLTCN or *SVRCN.

Short retry count (SHORTRTY)

Specifies the short retry count for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of attempts that are made to establish a connection to the remote machine, at intervals specified by SHORTTMR, before the (normally longer) LONGRTY and LONGTMR are used.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

short-retry-count
Specify a value in the range 0 through 999999999. A value of 0 means that no retries are allowed.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR, *CLTCN or *SVRCN.

Long retry interval (LONGTMR)

Specifies the long retry wait interval for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by SHORTRTY has been exhausted.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

long-retry-interval
Specify a value in the range 0 through 999999999.

Note: For implementation reasons, the maximum retry interval that can be used is 999999; values exceeding this are treated as 999999.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR, *CLTCN or *SVRCN.

Long retry count (LONGRTY)

Specifies the long retry count for a sender, server or cluster channel (*SDR, *SVR, *CLUSSDR or *CLUSRCVR) that is started automatically by the channel initiator. This defines the maximum number of further attempts that are made to connect to the remote machine, at intervals specified by LONGTMR, after the count specified by SHORTRTY has been exhausted. An error message is logged if the connection is not established after the defined number of attempts.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

long-retry-count
Specify a value in the range 0 through 999999999. A value of 0 means that no retries are allowed.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR, *CLTCN or *SVRCN.

Security exit (SCYEXIT)

Specifies the name of the program to be called as the security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

  Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.
- On receipt of a response to a security message flow.

  Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The security exit program is not invoked.

security-exit-name
Specify the name of the security exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

Security exit (CSCYEXIT)

Specifies the name of the program to be called as the client security exit. If a nonblank name is defined, the exit is invoked at the following times:

- Immediately after establishing a channel.

  Before any messages are transferred, the exit is given the opportunity to instigate security flows to validate connection authorization.
- On receipt of a response to a security message flow.

  Any security message flows received from the remote processor on the remote machine are passed to the exit.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the SYSTEM.DEF.CLNTCONN channel.

***NONE**
The client security exit program is not invoked.

security-exit-name
Specify the name of the client security exit program.

Security exit user data (SCYUSRDATA)

Specifies a maximum of 32 characters of user data that is passed to the channel security exit program.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The user data for the security exit is not specified.

security-exit-user-data
Specify the user data for the security exit program.

Send exit (SNDEXIT)

Specifies the entry point of the program to be called as the send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The send exit is not invoked.

send-exit-name
Specify the name of the send exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

Send exit (CSNDEXIT)

Specifies the entry point of the program to be called as the client send exit. If a nonblank name is defined, the exit is invoked immediately before data is sent out on the network. The exit is given the complete transmission buffer before it is transmitted; the contents of the buffer can be modified as required.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the SYSTEM.DEF.CLNTCONN channel.

***NONE**
The client send exit is not invoked.

send-exit-name
Specify the name of the client send exit program.

Send exit user data (SNDUSRDATA)

Specifies a maximum of 32 characters of user data that is passed to the send exit program.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The user data for the send exit program is not specified.

send-exit-user-data
Specify a maximum of 32 characters of user data for the send exit program.

Receive exit (RCVEXIT)

Specifies the entry point of the program to be called as the receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The receive exit program is not invoked.

receive-exit-name
Specify the name of the receive exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

Receive exit (CRCVEXIT)

Specifies the entry point of the program to be called as the client receive exit. If a nonblank name is defined, the exit is invoked before data received from the network is processed. The complete transmission buffer is passed to the exit and the contents of the buffer can be modified as required.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the SYSTEM.DEF.CLNTCONN channel.

***NONE**
The client receive exit program is not invoked.

receive-exit-name
Specify the name of the client receive exit program.

Receive exit user data (RCVUSRDATA)

Specifies user data that is passed to the receive exit.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The user data for the receive exit program is not specified.

receive-exit-user-data
Specify a maximum of 32 characters of user data for the receive exit program.

Message exit (MSGEXIT)

Specifies the entry point of the program to be called as the message exit. If a nonblank name is defined, the exit is invoked immediately after a message has been retrieved from the transmission queue. The exit is given the entire application message and message descriptor for modification.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The message exit program is not invoked.

message-exit-name
Specify the name of the message exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Message exit user data (MSGUSRDATA)

Specifies user data that is passed to the message exit program.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The user data for the message exit program is not specified.

message-exit-user-data
Specify a maximum of 32 characters of user data for the message exit program.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Message retry exit (MSGRTYEXIT)

Specifies the entry point of the program to be called as the message retry exit.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The message retry exit program is not invoked.

message-retry-exit-name
Specify the name of the message retry exit program.

library-name
Specify the name of the library that contains the exit program. This parameter must be present if an exit program name is specified.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Message retry exit data (MSGRTYDATA)

Specifies user data that is passed to the message retry exit program.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The user data for the message retry exit program is not specified.

message-retry-exit-user-data
Specify a maximum of 32 characters of user data for the message retry exit program.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Number of message retries (MSGRTYNBR)

Specifies the number of times the channel will retry before it decides it cannot deliver the message. This attribute controls the action of the MCA only if the message-retry exit name is blank, the value of MSGRTYNBR is passed to the exit for the exit's use, but the number of retries performed is controlled by the exit, and not by this attribute.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

message-retry-number
Specify a value in the range 0 through 999999999. A value of 0 signifies no retries will be performed.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Message retry interval (MSGRTYITV)

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

This attribute controls the action of the MCA only if the message-retry exit name is blank, the value of MSGRTYITV is passed to the exit for the exit's use, but the retry interval is controlled by the exit, and not by this attribute.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

message-retry-number
Specify a value in the range 0 through 999999999. A value of 0 signifies that the retry will be performed as soon as possible.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Convert message (CVTMSG)

Specifies whether the application data in the message should be converted before the message is transmitted.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel for the type of channel being created.

***YES** The application data in the message is converted before sending.

***NO** The application data in the message is not converted before sending.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR, *CLTCN or *SVRCN.

Put authority (PUTAUT)

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

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***DFT** No authority check is made before the message is put on the destination queue.

***CTX** The user identifier in the message context information is used to establish authority to put the message.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *SVR, *CLTCN, *SVRCN or *CLUSSDR.

Sequence number wrap (SEQNUMWRAP)

Specifies the maximum message sequence number. When the maximum is reached, sequence numbers wrap to start again at 1.

Note: The maximum message sequence number is not negotiable; the local and remote channels must wrap at the same number.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

sequence-number-wrap-value
Specify a value in the range 100 through 999999999.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Maximum message length (MAXMSGLEN)

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

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

maximum-message-length
Specify a value in the range 0 through 104857600. A value of 0 signifies that the maximum length is unlimited.

Heartbeat interval (HRTBTINTVL)

Specifies The time, in seconds, between heartbeat flows passed from the sending MCA when there are no messages on the transmission queue. The heartbeat exchange gives the receiving MCA the opportunity to quiesce the channel.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

heart-beat-interval
Specify a value in the range 0 through 999999999. A value of 0 means that no heartbeat exchanges are to take place.

Note: For implementation reasons, the maximum heartbeat interval that can be used is 999999; values exceeding this are treated as 999999.

Non Persistent Message Speed (NPMSPEED)

Specifies whether the channel supports Fast Non Persistant Messages.

The possible values are:

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

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

***NORMAL**
The channel does not support fast non persistant messages.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Cluster Name (CLUSTER)

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

This parameter is valid only for *CLUSSDR and *CLUSRCVR channels. If the CLUSNL parameter is non-blank, this parameter must be blank.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
No cluster name is specififed.

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

Cluster Name List (CLUSNL)

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

This parameter is valid only for *CLUSSDR and *CLUSRCVR channels. If the CLUSTER parameter is non-blank, this parameter must be blank.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
No cluster namelist is specififed.

cluster-name-list
The name of the namelist specifying a list of clusters to which the channel belongs. The maximum length is 48 characters conforming to the rules for naming MQ objects.

Network Connection Priority (NETPRTY)

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

This parameter is valid only for *CLUSRCVR channels.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

network-connection-priority
Specify a value in the range 0 through 9; 0 is the lowest priority.

SSL CipherSpec (SSLCIPH)

SSLCIPH specifies the CipherSpec used in SSL channel negotiation. The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

cipherspec
The name of the CipherSpec.

SSL Client Authentication (SSLCAUTH)

SSLCAUTH specifies whether the channel should carry out client authentication over SSL. The parameter is used only for channels with SSLCIPH specified.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***REQUIRED**
Client authentication is required.

***OPTIONAL**
Client authentication is optional.

This parameter cannot be specified for channel types (CHLTYPE) *SDR, *CLTCN or *CLUSSDR.

SSL Peer name (SSLPEER)

SSLPEER specifies the X500 peer name used in SSL channel negotiation. The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

x500peername
The X500 peer name to use.

Local communication address (LOCLADDR)

Specifies the local communication address for the channel.

This parameter is only valid for *SDR, *SVR, *RQSTR, *CLUSSDR, *CLUSRCVR and *CLTCN channels.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
The connection is blank.

local-address
Only valid for transport type TCP/IP. Specify the optional IP address and optional port or port range used for outbound TCP/IP communications. The format is LOCLADDR([ip-addr][(low-port[,high-port])]).

Batch Heartbeat Interval (BATCHHB)

The time in milliseconds used to determine whether batch heartbeating occurs on this channel. Batch heartbeating allows sender-type channels to determine whether the remote channel instance is still active before going in-doubt. A batch heartbeat will occur if a sender-type channel has not communicated with the remote channel within the specified time.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

batch-heartbeat-interval
Specify a value in the range 0 through 999999999. A value of 0 indicates that batch heartbeating is not to be used.

Note: For implementation reasons, the maximum batch heartbeat interval that can be used is 999999; values exceeding this are treated as 999999.

This parameter cannot be specified for channel types (CHLTYPE) *RCVR, *RQSTR, *CLTCN or *SVRCN.

Task user identifier (USERID)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

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

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

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
No user identifier is specified.

user-identifier
Specify the task user identifier.

Password (PASSWORD)

This is used by the message channel agent when attempting to initiate a secure LU 6.2 session with a remote message channel agent.

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

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

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
No password is specified.

Password
Specify the password.

Keep Alive Interval (KAINT)

Specifies the Keep Alive timing interval for this channel.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel for the type of channel being created.

***AUTO**
The Keep Alive interval is calculated based upon the negotiated heartbeat value as follows:

- If the negotiated HBINT is greater than 0, Keep Alive interval is set to that value plus 60 seconds.
- If the negotiated HBINT is 0, the value used is that specified by the KEEPALIVEOPTIONS statement in the TCP profile configuration data set.

keep-alive-interval
Specify a value in the range 0 through 99999.

Header Compression (COMPHDR)

The list of header data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (*SDR, *SVR, *CLUSSDR, *CLUSRCVR and *CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
No header data compression is performed.

***SYSTEM**
Header data compression is performed.

Message Compression (COMPMSG)

The list of message data compression techniques supported by the channel.

For channel types sender, server, cluster sender, cluster receiver and client connection (*SDR, *SVR, *CLUSSDR, *CLUSRCVR and *CLTCN) the values specified are in order of preference with the first compression technique supported by the remote end of the channel being used.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***NONE**
No message data compression is performed.

***RLE** Message data compression is performed using run-length encoding.

***ZLIBFAST**
Message data compression is performed using the zlib compression technique. A fast compression time is preferred.

***ZLIBHIGH**
Message data compression is performed using the zlib compression technique. A high level of compression is preferred.

***ANY** Any compression technique supported by the queue manager can be used. Only valid for channel types Receiver, Requester and Server-Connection.

Channel Monitoring (MONCHL)

Controls the collection of online monitoring data.

Online monitoring data is not collected when the queue manager attribute MONCHL is set to *NONE.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***QMGR**
The collection of Online Monitoring Data is inherited from the setting of the queue manager attribute MONCHL.

***NONE**
Online Monitoring Data collection for this channel is switched off.

***LOW** Monitoring data collection is turned on with a low ratio of data collection.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for a channel type (CHLTYPE) of *CLTCN.

Channel Statistics (STATCHL)

Controls the collection of statistics data.

Statistics data is not collected when the queue manager attribute STATCHL is set to *NONE.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***QMGR**
Statistics data collection is based upon the setting of the queue manager attribute STATCHL.

***NONE**
Statistics data collection for this channel is switched off.

***LOW** Statistics data collection is turned on with a low ratio of data collection.

***MEDIUM**
Statistics data collection is turned on with a moderate ratio of data collection.

***HIGH**
Statistics data collection is turned on with a high ratio of data collection.

This parameter cannot be specified for channel types (CHLTYPE) *CLTCN or *SVRCN.

Cluster Workload Rank (CLWLRANK)

Specifies the cluster workload rank of the channel.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

cluster-workload-rank
The cluster workload rank of the channel in the range 0 through 9.

Cluster Workload Priority (CLWLPRTY)

Specifies the cluster workload priority of the channel.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

cluster-workload-rank
The cluster workload priority of the channel in the range 0 through 9.

Cluster Channel Weight (CLWLWGHT)

Specifies the cluster workload weight of the channel.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

cluster-workload-rank
The cluster workload weight of the channel in the range 1 through 99.

Sharing Conversations (SHARECNV)

Specifies the maximum the number of conversations which can be shared over a particular TCP/IP client channel instance (socket).

This parameter is valid for channels with CHLTYPE defined as *CLTCN or *SVRCN.

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

0 Specifies no sharing of conversations over a TCP/IP socket. The channel instance runs in a mode prior to that of WebSphere MQ Version 7.0, with regard to:

- Administrator stop-quiesce
- Heartbeating
- Read ahead

1 Specifies no sharing of conversations over a TCP/IP socket. Client heartbeating and read ahead are available, whether in an MQGET call or not, and channel quiescing is more controllable.

shared-conversations
The number of shared conversations in the range 2 through 999999999.

Note: If the client-connection SHARECNV value does not match the server-connection SHARECNV value, the lower of the two values is used.

Property Control (PROPCTL)

Specifies what happens to properties of messages when the message is about to be sent to a V6 or prior queue manager (a queue manager that does not understand the concept of a property descriptor).

The possible values are:

***SYSDFTCHL**
The value of this attribute is taken from the system default channel of the specified type.

***COMPAT**
If the message contains a property with a prefix of ″mcd.″, ″jms.″, ″usr.″ or ″mqext.″ then all optional message properties, except those in the message descriptor (or extension) will be placed in one or more MQRFH2 headers in the message data before the message is sent to the remote queue manager.

***NONE**
All properties of the message, except those in the message descriptor (or extension), will be removed from the message before the message is sent to the remote queue manager.

***ALL** All properties of the message will be included with the message when it is sent to the remote queue manager. The properties, except those in the message descriptor (or extension), will be placed in one or more MQRFH2 headers in the message data.

Maximum Instances (MAXINST)

Specifies the maximum number of simultaneous instances of an individual server-connection channel.

The possible values are:

***SYSDFT**
The value of this attribute is taken from the system default channel of the specified type.

maximum-instances
The maximum number of simultaneous instances of the channel in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

Maximum Instances Per Client (MAXINSTC)

Specifies the maximum number of simultaneous instances of an individual server-connection channel which can be started from a single client.

The possible values are:

***SYSDFT**
The value of this attribute is taken from the system default channel of the specified type.

maximum-instances-per-client
The maximum number of simultaneous instances of the channel which can be in the started from a single client in the range 0 through 99999999.

A value of zero prevents all client access. If the value is reduced below the number of instances of the server connection channel currently running from individual clients, the running channels will not be affected, but new instances will not be able to start until sufficient existing ones have ceased to run.

Client Channel Weight (CLNTWGHT)

The client channel weighting attribute is used so client channel definitions can be selected at random based on their weighting when more than one suitable definition is available.

The possible values are:

***SYSDFT**
The value of this attribute is taken from the system default channel of the specified type.

client-channel-weight
The client channel weight in the range 0 through 99.

Client Channel Affinity (AFFINITY)

The channel affinity attribute is used so client applications that connect multiple times using the same queue manager name can choose whether to use the same client channel definition for each connection.

The possible values are:

***SYSDFT**
The value of this attribute is taken from the system default channel of the specified type.

***PREFERRED**
The first connection in a process reading a CCDT creates a list of applicable definitions based on the weighting with any applicable CLNTWGHT(0) definitions first and in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful non CLNTWGHT(0) definitions are moved to the end of the list. CLNTWGHT(0) definitions remain at the start of the list and are selected first for each connection.

***NONE**
The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the weighting with any applicable CLNTWGHT(0) definitions selected first in alphabetical order.

Examples

None

Error messages

Unknown

Create MQ Listener (CRTMQMLSR)

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

The Create MQ Listener (CRTMQMLSR) command creates a new MQ listener definition, specifying those attributes that are to be different from the default.

Parameters

Keyword	Description	Choices	Notes
LSRNAME	Listener name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
REPLACE	Replace	***NO**, *YES	Optional, Positional 3
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTLSR**	Optional, Positional 4
CONTROL	Listener control	***SYSDFTLSR**, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 5
PORT	Port number	0-65535, ***SYSDFTLSR**	Optional, Positional 6
IPADDR	IP Address	*Character value*, *BLANK, ***SYSDFTLSR**	Optional, Positional 7
BACKLOG	Listener backlog	0-999999999, ***SYSDFTLSR**	Optional, Positional 8

Listener name (LSRNAME)

The name of the new MQ listener definition to be created.

The possible values are:

listener-name
Specify the name of the listener definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Replace (REPLACE)

If a listener definition with the same name already exists, this specifies whether it is to be replaced.

The possible values are:

***NO** This definition does not replace any existing listener definition with the same name. The command fails if the named listener definition already exists.

***YES** Replace the existing listener definition. If there is no definition with the same name, a new definition is created.

Text 'description' (TEXT)

Specifies text that briefly describes the listener definition.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SYSDFTLSR**
The value of this attribute is taken from the system default listener.

***BLANK**
The text is set to a blank string.

description
Specify the new descriptive information.

Listener control (CONTROL)

Whether the listener starts automatically when the queue manager is started.

The possible values are:

***SYSDFTLSR**
The value for this attribute is taken from the system default listener.

***MANUAL**
The listener is not automatically started or stopped.

***QMGR**
The listener is started and stopped as the queue manager is started and stopped.

***STARTONLY**
The listener is started as the queue manager is started, but is not requested to stop when the queue manager is stopped.

Port number (PORT)

The port number to be used by the listener.

The possible values are:

***SYSDFTLSR**
The value for this attribute is taken from the system default listener.

port-number
The port number to be used.

IP Address (IPADDR)

The IP address to be used by the listener.

The possible values are:

***SYSDFTLSR**
The value for this attribute is taken from the system default listener.

ip-addr
The IP address to be used.

Listener backlog (BACKLOG)

The number of concurrent connection requests the listener supports.

The possible values are:

***SYSDFTLSR**
The value for this attribute is taken from the system default listener.

backlog
The number of concurrent connection requests supported.

Examples

None

Error messages

Unknown

Create MQ Namelist (CRTMQMNL)

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

The Create MQ Namelist (CRTMQMNL) command creates a new MQ namelist. A namelist is an MQ object that contains a list of other MQ objects. Typically, namelists are used by applications, for example trigger monitors, where they are used to identify a group of queues. A namelist is maintained independently of applications, therefore you can update it without stopping any of the applications that use it.

Parameters

Keyword	Description	Choices	Notes
NAMELIST	Namelist	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
REPLACE	Replace	***NO**, *YES	Optional, Positional 3
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTNL**	Optional, Positional 4
NAMES	List of Names	Values (up to 256 repetitions): *Character value*, *BLANKS, ***SYSDFTNL**, *NONE	Optional, Positional 5

Namelist (NAMELIST)

The name of the namelist to be created.

namelist
Specify the name of the namelist. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used.

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

Replace (REPLACE)

Specifies whether the new namelist should replace an existing namelist with the same name.

***NO** Do not replace the existing namelist. The command fails if the named namelist already exists.

***YES** Replace the existing namelist. If there is no namelist with the same name, a new namelist is created.

Text 'description' (TEXT)

Specifies text that briefly describes the namelist.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

***SYSDFTNL**
The value of the attribute is taken from the system default namelist.

description
Specify no more than 64 characters enclosed in apostrophes.

List of Names (NAMES)

List of names. This is the list of names to be created. The names can be of any type, but must conform to the rules for naming MQ objects.

***SYSDFTNL**
The value of the attribute is taken from the system default namelist.

namelist
The list to create. An empty list is valid.

Examples

None

Error messages

Unknown

Create MQ Process (CRTMQMPRC)

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

The Create MQ Process (CRTMQMPRC) command creates a new MQ process definition, specifying those attributes that are different from the default.

Parameters

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

Process name (PRCNAME)

The name of the new MQ process definition to be created.

The possible values are:

process-name
Specify the name of the new MQ process definition. The name can contain up to 48 characters.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Replace (REPLACE)

If a process definition with the same name already exists, this specifies whether it is replaced.

The possible values are:

***NO** This definition does not replace any existing process definition with the same name. The command fails if the named process definition already exists.

***YES** Replace the existing process definition. If there is no definition with the same name, a new definition is created.

Text 'description' (TEXT)

Specifies text that briefly describes the process definition.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

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

***BLANK**
The text is set to a blank string.

description
Specify the new descriptive information.

Application type (APPTYPE)

The type of application started.

The possible values are:

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

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

***MVS** Represents an MVS application.

***IMS** Represents an IMS application.

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

***DOS** Represents a DOS application.

***UNIX**
Represents a UNIX application.

***QMGR**
Represents a queue manager.

***OS400**
Represents an i5/OS application.

***WINDOWS**
Represents a Windows application.

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

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

***VMS** Represents a VMS application.

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

***VOS** Represents a VOS application.

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

***XCF** Represents an XCF application.

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

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

***BROKER**
Represents a broker application.

***JAVA** Represents a Java application.

***DQM**
Represents a DQM application.

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

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

Application identifier (APPID)

Application identifier. This is the name of the application to be started, on the platform for which the command is processing. It is typically a program name and library name.

The possible values are:

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

application-id
The maximum length is 256 characters.

User data (USRDATA)

A character string that contains user information pertaining to the application, as defined by APPID, to start.

The possible values are:

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

***NONE**
The user data is blank.

user-data
Specify up to 128 characters of user data.

Environment data (ENVDATA)

A character string that contains environment information pertaining to the application, as defined by APPID, to start.

The possible values are:

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

***NONE**
The environment data is blank.

environment-data
The maximum length is 128 characters.

Examples

None

Error messages

Unknown

Create MQ Queue (CRTMQMQ)

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

The Create MQ Queue (CRTMQMQ) command creates a queue definition with the specified attributes. All attributes that are not specified are set to the default value for the type of queue that is created.

Parameters

Keyword	Description	Choices	Notes
QNAME	Queue name	*Character value*	Required, Key, Positional 1
QTYPE	Queue type	*ALS, *LCL, *MDL, *RMT	Required, Key, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 3
REPLACE	Replace	***NO** , *YES	Optional, Positional 4
TEXT	Text 'description'	*Character value*, *BLANK, ***SYSDFTQ**	Optional, Positional 5
PUTENBL	Put enabled	***SYSDFTQ** , *NO, *YES	Optional, Positional 6
DFTPTY	Default message priority	0-9, ***SYSDFTQ**	Optional, Positional 7
DFTMSGPST	Default message persistence	***SYSDFTQ** , *NO, *YES	Optional, Positional 8
PRCNAME	Process name	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 9
TRGENBL	Triggering enabled	***SYSDFTQ** , *NO, *YES	Optional, Positional 10
GETENBL	Get enabled	***SYSDFTQ** , *NO, *YES	Optional, Positional 11
SHARE	Sharing enabled	***SYSDFTQ** , *NO, *YES	Optional, Positional 12
DFTSHARE	Default share option	***SYSDFTQ** , *NO, *YES	Optional, Positional 13
MSGDLYSEQ	Message delivery sequence	***SYSDFTQ** , *PTY, *FIFO	Optional, Positional 14
HDNBKTCNT	Harden backout count	***SYSDFTQ** , *NO, *YES	Optional, Positional 15
TRGTYPE	Trigger type	***SYSDFTQ** , *FIRST, *ALL, *DEPTH, *NONE	Optional, Positional 16
TRGDEPTH	Trigger depth	1-999999999, ***SYSDFTQ**	Optional, Positional 17

Keyword	Description	Choices	Notes
TRGMSGPTY	Trigger message priority	0-9, ***SYSDFTQ**	Optional, Positional 18
TRGDATA	Trigger data	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 19
RTNITV	Retention interval	0-999999999, ***SYSDFTQ**	Optional, Positional 20
MAXDEPTH	Maximum queue depth	0-999999999, ***SYSDFTQ**	Optional, Positional 21
MAXMSGLEN	Maximum message length	0-104857600, ***SYSDFTQ**	Optional, Positional 22
BKTTHLD	Backout threshold	0-999999999, ***SYSDFTQ**	Optional, Positional 23
BKTQNAME	Backout requeue name	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 24
INITQNAME	Initiation queue	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 25
USAGE	Usage	***SYSDFTQ** , *NORMAL, *TMQ	Optional, Positional 26
DFNTYPE	Definition type	***SYSDFTQ** , *TEMPDYN, *PERMDYN	Optional, Positional 27
TGTQNAME	Target queue	*Character value*, ***SYSDFTQ**	Optional, Positional 28
RMTQNAME	Remote queue	*Character value*, ***SYSDFTQ** , *NONE	Optional, Positional 29
RMTMQMNAME	Remote Message Queue Manager	*Character value*, ***SYSDFTQ**	Optional, Positional 30
TMQNAME	Transmission queue	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 31
HIGHTHLD	Queue depth high threshold	0-100, ***SYSDFTQ**	Optional, Positional 32
LOWTHLD	Queue depth low threshold	0-100, ***SYSDFTQ**	Optional, Positional 33
FULLEVT	Queue full events enabled	***SYSDFTQ** , *NO, *YES	Optional, Positional 34
HIGHEVT	Queue high events enabled	***SYSDFTQ** , *NO, *YES	Optional, Positional 35
LOWEVT	Queue low events enabled	***SYSDFTQ** , *NO, *YES	Optional, Positional 36
SRVITV	Service interval	0-999999999, ***SYSDFTQ**	Optional, Positional 37
SRVEVT	Service interval events	***SYSDFTQ** , *HIGH, *OK, *NONE	Optional, Positional 38
DISTLIST	Distribution list support	***SYSDFTQ** , *NO, *YES	Optional, Positional 39
CLUSTER	Cluster Name	*Character value*, ***SYSDFTQ** , *NONE	Optional, Positional 40

Keyword	Description	Choices	Notes
CLUSNL	Cluster Name List	*Character value*, *NONE, ***SYSDFTQ**	Optional, Positional 41
DEFBIND	Default Binding	***SYSDFTQ** , *OPEN, *NOTFIXED	Optional, Positional 42
CLWLRANK	Cluster Workload Rank	0-9, ***SYSDFTQ**	Optional, Positional 43
CLWLPRTY	Cluster Workload Priority	0-9, ***SYSDFTQ**	Optional, Positional 44
CLWLUSEQ	Cluster workload queue use	***SYSDFTQ** , *QMGR, *LOCAL, *ANY	Optional, Positional 45
MONQ	Queue Monitoring	***SYSDFTQ** , *QMGR, *OFF, *LOW, *MEDIUM, *HIGH	Optional, Positional 46
STATQ	Queue Statistics	***SYSDFTQ** , *QMGR, *OFF, *ON	Optional, Positional 47
ACCTQ	Queue Accounting	***SYSDFTQ** , *QMGR, *OFF, *ON	Optional, Positional 48
NPMCLASS	Non Persistent Message Class	***SYSDFTQ** , *NORMAL, *HIGH	Optional, Positional 49
MSGREADAHD	Message Read Ahead	***SYSDFTQ** , *DISABLED, *NO, *YES	Optional, Positional 50
DFTPUTRESP	Default Put Response	*SYSDFTQ, ***SYNC** , *ASYNC	Optional, Positional 51
PROPCTL	Property Control	***SAME** , *COMPAT, *NONE, *ALL, *FORCE	Optional, Positional 52
TARGTYPE	Target Type	***SAME** , *QUEUE, *TOPIC	Optional, Positional 53

Queue name (QNAME)

Specifies the name of the queue definition. Queue names must be unique. If a queue definition with this name already exists, you must specify REPLACE(*YES).

The name can contain up to 48 characters.

Note: The field length is 48 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

The possible values are:

queue-name
Specify the name of the new queue.

Queue type (QTYPE)

Specifies the type of queue that is to be created.

If the queue already exists, REPLACE(*YES) must be specified, and the value specified by QTYPE must be the type of the existing queue.

The possible values are:

***ALS** An alias queue.

***LCL** A local queue.

***RMT** A remote queue.

***MDL** A model queue.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Replace (REPLACE)

Specifies whether the new queue will replace an existing queue definition with the same name and type.

The possible values are:

***NO** Do not replace the existing queue. The command fails if the named queue already exists.

***YES** Replace the existing queue definition with the attributes of the FROMQ and the specified attributes.

The command will fail if an application has the Queue open or the USAGE attribute is changed.

Note: If the queue is a local queue, and a queue with the same name already exists, any messages already on that queue are retained.

Text 'description' (TEXT)

Specifies text that briefly describes the queue definition.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***BLANK**
The text is set to a blank string.

description
Specify no more than 64 characters enclosed in apostrophes.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Put enabled (PUTENBL)

Specifies whether messages can be put on the queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NO** Messages cannot be added to the queue.

***YES** Messages can be added to the queue by authorized applications.

Default message priority (DFTPTY)

Specifies the default priority of messages put on the queue.

The possible values are:

***SYSDFTQ**
The value of this attribute taken from the system default queue of the specified type.

priority-value
Specify a value ranging from 0 through 9.

Default message persistence (DFTMSGPST)

Specifies the default for message-persistence on the queue. Message persistence determines whether or not messages are preserved across restarts of the queue manager.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NO** By default, messages are lost across a restart of the queue manager.

***YES** By default, messages are preserved across a restart of the queue manager.

Process name (PRCNAME)

Specifies the local name of the MQ process that identifies the application that should be started when a trigger event occurs.

The process does not have to be available when the queue is created, but it must be available for a trigger event to occur.

The possible values are:

***SYSDFTQ**
The value of this attribute taken from the system default queue of the specified type.

***NONE**
No process is specified.

process-name
Specify the name of the process.

Triggering enabled (TRGENBL)

Specifies whether trigger messages are written to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NO** Do not write trigger messages to the initiation queue.

***YES** Triggering is active; trigger messages are written to the initiation queue.

Get enabled (GETENBL)

Specifies whether applications are to be permitted to get messages from this queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NO** Applications cannot retrieve messages from the queue.

***YES** Suitably authorized applications can retrieve messages from the queue.

Sharing enabled (SHARE)

Specifies whether multiple instances of applications can open this queue for input.

The possible values are:

***SYSDFTQ**
The value of this attribute is from the system default queue of the specified type.

***NO** Only a single application instance can open the queue for input.

***YES** More than one application instance can open the queue for input.

Default share option (DFTSHARE)

Specifies the default share option for applications opening this queue for input.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NO** The open request is for exclusive use of the queue for input.

***YES** The open request is for shared use of the queue for input.

Message delivery sequence (MSGDLYSEQ)

Specifies the message delivery sequence.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***PTY** Messages are delivered in first-in-first-out (FIFO) order within priority.

***FIFO** Messages are delivered in FIFO order regardless of priority.

Harden backout count (HDNBKTCNT)

Specifies whether the count of backed out messages should be saved (hardened) across restarts of the message queue manager.

Note: On WebSphere MQ for i5/OS the count is ALWAYS hardened, regardless of the setting of this attribute.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NO** The backout count is not hardened.

***YES** The backout count is hardened.

Trigger type (TRGTYPE)

Specifies the condition that initiates a trigger event. When the condition is true, a trigger message is sent to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***FIRST**
When the number of messages on the queue goes from zero to one.

***ALL** Every time a message arrives on the queue.

***DEPTH**
When the number of messages on the queue equals the value of the TRGDEPTH attribute.

***NONE**
No trigger messages are written.

Trigger depth (TRGDEPTH)

Specifies, for TRIGTYPE(*DEPTH), the number of messages that initiate a trigger message to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

depth-value
Specify a value ranging from 1 through 999999999.

Trigger message priority (TRGMSGPTY)

Specifies the minimum priority that a message must have before it can produce, or be counted for, a trigger event.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

priority-value
Specify a value ranging from 0 through 9.

Trigger data (TRGDATA)

Specifies up to 64 characters of user data that the queue manager includes in the trigger message. This data is made available to the monitoring application that processes the initiation queue and to the application started by the monitor.

Note: An application program can issue a call to MQSET to change the value of this attribute.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NONE**
No trigger data is specified.

trigger-data
Specify up to 64 characters enclosed in apostrophes. For a transmission queue you can use this parameter to specify the name of the channel to be started.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Retention interval (RTNITV)

Specifies the retention interval. This interval is the number of hours for which the queue might be needed, based on the date and time when the queue was created.

This information is available to a housekeeping application or an operator and can be used to determine when a queue is no longer required.

Note: The message queue manager does not delete queues, nor does it prevent your queues from being deleted if their retention interval has not expired. It is your responsibility to take any required action.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

interval-value
Specify a value ranging from 0 through 999999999.

Maximum queue depth (MAXDEPTH)

Specifies the maximum number of messages allowed on the queue. However, other factors can cause the queue to be treated as full; for example, it appears to be full if there is no storage available for a message.

Note: If this value is subsequently reduced by using the CHGMQMQ command, any messages that are on the queue remain intact even if they cause the new maximum to be exceeded.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

depth-value
Specify a value ranging from 0 through 999999999.

Maximum message length (MAXMSGLEN)

Specifies the maximum length for messages on the queue.

Note: If this value is subsequently reduced by using the CHGMQMQ command, any messages that are on the queue remain intact even if they exceed the new maximum length.

Applications might use the value of this attribute to determine the size of buffer they need to retrieve messages from the queue. Therefore change the value only if you know this will not cause an application to operate incorrectly.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified queue type.

length-value
Specify a value ranging from 0 through 104 857 600.

Backout threshold (BKTTHLD)

Specifies the backout threshold. Apart from allowing this attribute to be queried, WebSphere MQ for i5/OS takes no action based on the value of the attribute.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified queue type.

threshold-value
Specify a value ranging from 0 through 999999999.

Backout requeue name (BKTQNAME)

Specifies the backout-queue name. Apart from allowing this attribute to be queried, WebSphere MQ for i5/OS takes no action based on the value of the attribute.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified queue type.

***NONE**
No backout queue is specified.

backout-queue-name
Specify the backout queue name.

Initiation queue (INITQNAME)

Specifies the name of the initiation queue.

Note: The initiation queue must be on the same instance of a message queue manager.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified queue type.

***NONE**
No initiation queue is specified.

initiation-queue-name
Specify the initiation queue name.

Usage (USAGE)

Specifies whether the queue is for normal usage, or for transmitting messages to a remote message queue manager.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified queue type.

***NORMAL**
Normal usage (the queue is not a transmission queue)

***TMQ** The queue is a transmission queue that is used to hold messages destined for a remote message queue manager. If the queue is intended for use in situations where a transmission queue name is not explicitly specified, the queue name must be the same as the name of the remote message queue manager. For further information, see the WebSphere MQ Intercommunication.

Definition type (DFNTYPE)

Specifies the type of dynamic queue definition that is created when an application issues an MQOPEN API call with the name of this model queue specified in the object descriptor.

Note: This parameter only applies to a model queue definition.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***TEMPDYN**
Creates a temporary dynamic queue. Do not specify with a DEFMSGPST value of *YES.

***PERMDYN**
Creates a permanent dynamic queue.

Target queue (TGTQNAME)

Specifies the name of the target object for which this queue is an alias.

The object can be a local or remote queue, a topic or a message queue manager.

Do not leave this field blank. If you do so, it is possible that you will create an alias queue, that has to be subsequently modified, by the addition of a TGTNAME.

When a message queue manager name is specified, it identifies the message queue manager that handles the messages posted to the alias queue. You can specify either the local message queue manager or a transmission queue name.

Note: The target object does not need to exist at this time but it must exist when a process attempts to open the alias queue.

The possible values are:

***SYSDFTQ**
The name of the target objecy is taken from the SYSTEM.DEFAULT.ALIAS.QUEUE.

target-object-name
Specify the name of the target object.

Remote queue (RMTQNAME)

Specifies the name of the remote queue. That is, the local name of the remote queue as defined on the queue manager specified by RMTMQMNAME.

If this definition is used for a queue manager alias definition, RMTQNAME must be blank when the open occurs.

If this definition is used for a reply-to alias, this name is the name of the queue that is to be the reply-to queue.

The possible values are:

***SYSDFTQ**
The name of the remote queue is taken from the SYSTEM.DEFAULT.REMOTE.QUEUE.

***NONE**
No remote-queue name is specified (that is, the name is blank). This can be used if the definition is a queue manager alias definition.

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

Note: The name is not checked to ensure that it contains only those characters normally allowed for queue names

Remote Message Queue Manager (RMTMQMNAME)

Specifies the name of the remote queue manager on which the queue RMTQNAME is defined.

If an application opens the local definition of a remote queue, RMTMQMNAME must not be the name of the connected queue manager. If TMQNAME is blank there must be a local queue of this name, which is to be used as the transmission queue.

If this definition is used for a queue manager alias, RMTMQMNAME is the name of the queue manager, which can be the name of the connected queue manager. Otherwise, if TMQNAME is blank, when the queue is opened there must be a local queue of this name, with USAGE(*TMQ) specified, which is to be used as the transmission queue.

If this definition is used for a reply-to alias, this name is the name of the queue manager that is to be the reply-to queue manager.

The possible values are:

***SYSDFTQ**
The name of the remote queue manager is taken from the SYSTEM.DEFAULT.REMOTE.QUEUE.

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

Note: Ensure this name contains only those characters normally allowed for queue manager names.

Transmission queue (TMQNAME)

Specifies the local name of the transmission queue to be used for messages destined for the remote queue, for either a remote queue or for a queue manager alias definition.

If TMQNAME is blank, a queue with the same name as RMTMQMNAME is used as the transmission queue.

This attribute is ignored if the definition is being used as a queue manager alias and RMTMQMNAME is the name of the connected queue manager.

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

The possible values are:

***SYSDFTQ**
The transmission queue name is taken from the SYSTEM.DEFAULT.REMOTE.QUEUE.

***NONE**
No specific transmission queue name is defined for this remote queue. The value of this attribute is set to all blanks.

transmission-queue-name
Specify the transmission queue name.

Queue depth high threshold (HIGHTHLD)

Specifies the threshold against which the queue depth is compared to generate a queue depth high event.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

threshold-value
Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

Queue depth low threshold (LOWTHLD)

Specifies the threshold against which the queue depth is compared to generate a queue depth low event.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

threshold-value
Specify a value ranging from 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

Queue full events enabled (FULLEVT)

Specifies whether queue full events are generated.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NO** Queue Full events are not generated.

***YES** Queue Full events are generated.

Queue high events enabled (HIGHEVT)

Specifies whether queue depth high events are generated.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NO** Queue Depth High events are not generated.

***YES** Queue Depth High events are generated.

Queue low events enabled (LOWEVT)

Specifies whether queue depth low events are generated.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NO** Queue Depth Low events are not generated.

***YES** Queue Depth Low events are generated.

Service interval (SRVITV)

Specifies the service interval. This interval is used for comparison to generate service interval high and service interval OK events.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

interval-value
Specify a value ranging from 0 through 999999999. The value is in units of milliseconds.

Service interval events (SRVEVT)

Specifies whether service interval high or service interval OK events are generated.

A service interval high event is generated when a check indicates that no messages have been retrieved from the queue for the time indicated by the SRVITV parameter as a minimum.

A service interval OK event is generated when a check indicates that messages have been retrieved from the queue within the time indicated by the SRVITV parameter.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***HIGH**
Service Interval High events are generated.

***OK** Service Interval OK events are generated.

***NONE**
No service interval events are generated.

Distribution list support (DISTLIST)

Specifies whether the queue supports distribution lists.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NO** Distribution Lists are not supported.

***YES** Distribution Lists are supported.

Cluster Name (CLUSTER)

The name of the cluster to which the queue belongs.

Changes to this parameter do not affect instances of the queue that are already open.

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

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

cluster-name
Only one of the resultant values of CLUSTER or CLUSNL can be non-blank; you cannot specify a value for both.

Cluster Name List (CLUSNL)

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

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

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

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

Default Binding (DEFBIND)

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

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

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

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

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

Cluster Workload Rank (CLWLRANK)

Specifies the cluster workload rank of the queue.

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

cluster-workload-rank
Specify a value ranging from 0 through 9.

Cluster Workload Priority (CLWLPRTY)

Specifies the cluster workload priority of the queue.

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

cluster-workload-priority
Specify a value ranging from 0 through 9.

Cluster workload queue use (CLWLUSEQ)

Specifies the behaviour of an MQPUT when the target queue has both a local instance and at least one remote cluster instance. If the put originates from a cluster channel then this attribute does not apply.

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***QMGR**
The value is inherited from the Queue Manager CLWLUSEQ attribute.

***LOCAL**
The local queue will be the sole target of the MQPUT.

***ANY** The queue manager will treat such a local queue as another instance of the cluster queue for the purposes of workload distribution.

Queue Monitoring (MONQ)

Controls the collection of Online Monitoring Data.

Online Monitoring Data is not collected when the queue manager attribute MONQ is set to *NONE.

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***QMGR**
The collection of Online Monitoring Data is inherited from the setting of the queue manager attribute MONQ.

***OFF** Online Monitoring Data collection for this queue is switched off.

***LOW** Monitoring data collection is turned on with a low ratio of data collection.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection.

Queue Statistics (STATQ)

Controls the collection of statistics data.

Online monitoring data is not collected when the queue manager attribute STATQ is set to *NONE.

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***QMGR**
Statistics data collection is based upon the setting of the queue manager attribute STATQ.

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

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

Queue Accounting (ACCTQ)

Controls the collection of accouting data.

Accounting data is not collected when the queue manager attribute ACCTQ is set to *NONE.

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

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

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

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

Non Persistent Message Class (NPMCLASS)

Specifies the level of reliability for non-persistent messages put to this queue.

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***NORMAL**
Non-persistent messages put to this queue are only lost following a failure, or a queue manager shutdown. Non-persistent message put to this queue will be discarded in the event of a queue manager restart.

***HIGH**
Non-persistent messages put to this queue are not discarded in the event of a queue manager restart. Non-persistent messages put to this queue may still be lost in the event of a failure.

Message Read Ahead (MSGREADAHD)

Specifies whether non persistent messages are sent to the client ahead of an application requesting them.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***DISABLED**
Read ahead is disabled for this queue. Messages are not sent to the client ahead of an application requesting them regardless of whether read ahead is requested by the client application.

***NO** Non-persistent messages are not sent to the client ahead of an application requesting them. A maximum of one non-persistent message can be lost if the client ends abnormally.

***YES** Non-persistent messages are sent to the client ahead of an application requesting them. Non-persistent messages can be lost if the client ends abnormally or if the client application does not consume all the messages it is sent.

Default Put Response (DFTPUTRESP)

The default put response type (DFTPUTRESP) attribute specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO_RESPONSE_AS_Q_DEF option.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***SYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are issued as if MQPMO_SYNC_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application. This is the default value supplied with WebSphere MQ, but your installation might have changed it.

***ASYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are always issued as if MQPMO_ASYNC_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application; but an improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

Property Control (PROPCTL)

Specifies what happens to properties of messages that are retrieved from queues using the MQGET call when the MQGMO_PROPERTIES_AS_Q_DEF option is specified.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***COMPAT**
If the message contains a property with a prefix of ″mcd.″, ″jms.″, ″usr.″ or ″mqext.″ then all message properties are delivered to the application in an MQRFH2 header. Otherwise all properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

***NONE**
All properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application.

***ALL** All properties of the message, except those contained in the message descriptor (or extension), are contained in one or more MQRFH2 headers in the message data.

***FORCE**
Properties are always returned in the message data in an MQRFH2 header regardless of whether the application specifies a message handle.

Target Type (TARGTYPE)

Specifies the type of object to which the alias resolves.

The possible values are:

***SYSDFTQ**
The value of this attribute is taken from the system default queue of the specified type.

***QUEUE**
Queue object.

***TOPIC**
Topic object.

Examples

None

Error messages

Unknown

Create MQ Subscription (CRTMQMSUB)

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

The Create MQ Subscription (CRTMQMSUB) command creates a new MQ subsciption, specifying those attributes that are different from the default.

Parameters

Keyword	Description	Choices	Notes
SUBNAME	Subscription name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
REPLACE	Replace	***NO**, *YES	Optional, Positional 3
TOPICSTR	Topic string	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 4
TOPICOBJ	Topic object	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 5
DEST	Destination	*Character value*, ***SYSDFTSUB**	Optional, Positional 6
DESTMQM	Destination Queue Manager	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 7
DESTCRRLID	Destination Correlation Id	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 8
PUBACCT	Publish Accounting Token	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 9
PUBAPPID	Publish Application Id	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 10
SUBUSER	Subscription User Id	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 11
USERDATA	Subscription User Data	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 12
SELECTOR	Selector String	*Character value*, *NONE, ***SYSDFTSUB**	Optional, Positional 13

Keyword	Description	Choices	Notes
PSPROP	PubSub Property	***SYSDFTSUB**, *NONE, *COMPAT, *RFH2	Optional, Positional 14
DESTCLASS	Destination Class	***SYSDFTSUB**, *MANAGED, *PROVIDED	Optional, Positional 15
SUBSCOPE	Subscription Scope	***SYSDFTSUB**, *ALL, *QMGR	Optional, Positional 16
VARUSER	Variable User	***SYSDFTSUB**, *ANY, *FIXED	Optional, Positional 17
REQONLY	Request Publications	***SYSDFTSUB**, *YES, *NO	Optional, Positional 18
PUBPTY	Publish Priority	0-9, ***SYSDFTSUB**, *ASPUB, *ASQDEF	Optional, Positional 19
WSCHEMA	Wildcard Schema	***SYSDFTSUB**, *TOPIC, *CHAR	Optional, Positional 20
EXPIRY	Expiry Time	0-999999999, ***SYSDFTSUB**, *UNLIMITED	Optional, Positional 21

Subscription name (SUBNAME)

The name of the new MQ subscription to be created.

The possible values are:

subscription-name
Specify a maximum of 256 bytes for the subscription name.

Note: Subscription names of greater than 256 bytes can be speicifed using MQSC.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Replace (REPLACE)

If a subscription with the same name already exists, this specifies whether it is replaced.

The possible values are:

***NO** This subscription does not replace any existing subscription with the same name or subscription identifier. The command fails if the subscription already exists.

***YES** Replace the existing subscription. If there is no subscription with the same name or subscription identifier, a new subscription is created.

Topic string (TOPICSTR)

Specifies the topic string associated with this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

topic-string
Specify a maximum of 256 bytes for the topic string.

Note: Topic strings of greater than 256 bytes can be speicifed using MQSC.

Topic object (TOPICOBJ)

Specifies the topic object associated with this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

topic-object
Specifiy the name of the topic object.

Destination (DEST)

Specifies the destination queue for messages published to this subscription.

The possible values are:

destination-queue
Specify the name of the destination queue.

Destination Queue Manager (DESTMQM)

Specifies the destination queue manager for messages published to this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

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

Destination Correlation Id (DESTCRRLID)

Specifies the correlation identifier for messages published to this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

destination-correlation-identifier
Specify the 48 character hexadecimal string representing the 24 byte correlation identifier.

Publish Accounting Token (PUBACCT)

Specifies the accounting token for messages published to this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***NONE**
Messages are placed on the destination with an accounting token of MQACT_NONE.

publish-accounting-token
Specify the 64 character hexadecimal string representing the 32 byte publish accounting token.

Publish Application Id (PUBAPPID)

Specifies the publish application identity for messages published to this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***NONE**
No publish application identifier is specified.

publish-application-identifier
Specify the publish application identifier.

Subscription User Id (SUBUSER)

Specifies the user profile that 'owns' this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***NONE**
No user profile is specified.

user-profile
Specify the user profile.

Subscription User Data (USERDATA)

Specifies the user data associated with the subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***NONE**
No user data is specified.

user-data
Specify a maximum of 256 bytes for user data.

Note: User data of greater than 256 bytes can be speicifed using MQSC.

Selector String (SELECTOR)

Specifies the SQL 92 selector string to be applied to messages published on the named topic to select whether they are eligible for this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***NONE**
No selection string is specified.

selection-string
Specify a maximum of 256 bytes for selection string.

Note: Selection strings of greater than 256 bytes can be speicifed using MQSC.

PubSub Property (PSPROP)

Specifies the manner in which publish / subscribe related message properties are added to messages sent to this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***NONE**
Publish / subscribe properties are not added to the message.

***COMPAT**
Publish / subscribe properties are added to the message to maintain compatibility with V6 Publish / Subscribe.

***RFH2**
Publish / subscribe properties are added to the message within an RFH Version 2 header.

Destination Class (DESTCLASS)

Specifies whether this is a managed subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***MANAGED**
The destination is managed.

***PROVIDED**
The destination is a queue.

Subscription Scope (SUBSCOPE)

Specifies whether this subscription should be forwarded (as a proxy subscription) to other brokers, so that the subscriber will receive messages published at those other brokers.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***ALL** The subscription will be forwarded to all queue managers directly connected via a publish / subscribe collective or hierarchy.

***QMGR**
The subscription will only forward messages published on the topic within this queue manager.

Variable User (VARUSER)

Specifies whether user profiles other than the creator of the subscription can connect to it (subject to topic and destination authority checks).

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***ANY** Any user profiles can connect to the subsciption.

***FIXED**
Only the user profile that created the subscription can connect to it.

Request Publications (REQONLY)

Specifies whether the subscriber will poll for updates via MQSUBRQ API, or whether all publications are delivered to this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***YES** Publications are only delivered to this subscription in response to an MQSUBRQ API.

***NO** All publications on the topic are delivered to this subscription.

Publish Priority (PUBPTY)

Specifies the priority of the message sent to this subscription.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***ASPUB**
The priority of the message sent to this subscription is taken from that supplied in the published message.

***ASQDEF**
The priority of the message sent to this subscription is taken from the default priority of the queue defined as the destination.

priority-value
Specify a priority ranging from 0 through 9.

Wildcard Schema (WSCHEMA)

Specifies the schema to be used when interpreting wild card characters in the topic string.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***TOPIC**
Wildcard characters represent portions of the topic hierarchy.

***CHAR**
Wildcard characters represent portions of strings.

Expiry Time (EXPIRY)

Specifies the expiry time of the subscription. After a subscription's expiry time has elapsed, it becomes eligible to be discarded by the queue manager and will receive no further publications.

The possible values are:

***SYSDFTSUB**
The value of this attribute is taken from the system default subscription.

***UNLIMITED**
The subscription does not expire.

expiry-time
Specify an expiry time in tenths of a second ranging from 0 through 999999999.

Examples

None

Error messages

Unknown

Create MQ Service (CRTMQMSVC)

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

The Create MQ Service (CRTMQMSVC) command creates a new MQ service definition, specifying those attributes that are to be different from the default.

Parameters

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

Keyword	Description	Choices	Notes
CONTROL	Service control	***SYSDFTSVC**, *MANUAL, *QMGR, *STARTONLY	Optional, Positional 12

Service name (SVCNAME)

The name of the new MQ service definition.

The possible values are:

service-name
Specify the name of the service definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Replace (REPLACE)

If a service definition with the same name already exists, this specifies whether it is replaced.

The possible values are:

***NO** This definition does not replace any existing service definition with the same name. The command fails if the named service definition already exists.

***YES** Replace the existing service definition. If there is no definition with the same name, a new definition is created.

Text 'description' (TEXT)

Specifies text that briefly describes the service definition.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SYSDFTSVC**
The value of this attribute is taken from the system default service.

***BLANK**
The text is set to a blank string.

description
Specify the new descriptive information.

Start program (STRCMD)

The name of the program to run.

The possible values are:

***SYSDFTSVC**
The value of this attribute is taken from the system default service.

start-command
The name of the start command executable.

Start program arguments (STRARG)

The arguments passed to the program at startup.

The possible values are:

***SYSDFTSVC**
The value of this attribute is taken from the system default service.

***BLANK**
No arguments are passed to the start command.

start-command-arguments
The arguments passed to the start command.

End program (ENDCMD)

The name of the executable to run when the service is requested to stop.

The possible values are:

***SYSDFTSVC**
The value of this attribute is taken from the system default service.

***BLANK**
No end command is executed.

end-command
The name of the end command executable.

End program arguments (ENDARG)

The arguments passed to the end program when the service is requested to stop.

The possible values are:

***SYSDFTSVC**
The value of this attribute is taken from the system default service.

***BLANK**
No arguments are passed to the end command.

end-command-arguments
The arguments passed to the end command.

Standard output (STDOUT)

The path to a file to which the standard output of the service program is redirected.

The possible values are:

***SYSDFTSVC**
The value of this attribute is taken from the system default service.

***BLANK**
The standard output is discarded.

stdout-path
The standard output path.

Standard error (STDERR)

The path to a file to which the standard error of the service program is redirected.

The possible values are:

***SYSDFTSVC**
The value of this attribute is taken from the system default service.

***BLANK**
The standard error is discarded.

stderr-path
The standard error path.

Service type (TYPE)

Mode in which to run service.

The possible values are:

***SYSDFTSVC**
The value for this attribute is taken from the system default service.

***CMD** When started the command is executed but no status is collected or displayed.

***SVR** The status of the executable started will be monitored and displayed.

Service control (CONTROL)

Whether the service should be started automatically at queue manager start.

The possible values are:

***SYSDFTSVC**
The value for this attribute is taken from the system default service.

***MANUAL**
The service will not be automatically started or stopped.

***QMGR**
The service will be started and stopped as the queue manager is started and stopped.

***STARTONLY**
The service will be started as the queue manager is started, but will not be requested to stop when the queue manager is stopped.

Examples

None

Error messages

Unknown

Create MQ Topic (CRTMQMTOP)

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

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

Parameters

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

Topic name (TOPNAME)

The name of the new MQ topic object to be created.

The possible values are:

topic-name
Specify the name of the new MQ topic object. The name can contain up to 48 characters.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Replace (REPLACE)

If a topic object with the same name already exists, this specifies whether it is replaced.

The possible values are:

***NO** This object does not replace any existing topic object with the same name. The command fails if the named topic object already exists.

***YES** Replace the existing topic object. If there is no object with the same name, a new object is created.

Text 'description' (TEXT)

Specifies text that briefly describes the topic object.

Note: The field length is 64 bytes and the maximum number of characters is reduced if the system is using a double-byte character set (DBCS).

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

***BLANK**
The text is set to a blank string.

description
Specify the new descriptive information.

Topic string (TOPICSTR)

Specifies the topic string represented by this topic object definition.

The possible values are:

topic-string
Specify a maximum of 256 bytes for the topic string.

Note: Topic strings of greater than 256 bytes can be speicifed using MQSC.

Durable subscriptions (DURSUB)

Specifies whether applications are permitted to make durable subscriptions on this topic.

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

***ASPARENT**
Whether durable subscriptions can be made on this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***YES** Durable subscriptions can be made on this topic.

***NO** Durable subscriptions cannot be made on this topic.

Durable model queue (MGDDURMDL)

Specifies the name of the model queue to be used for durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

durable-model-queue
Specify the name of the model queue.

Non-durable model queue (MGDNDURMDL)

Specifies the name of the model queue to be used for non-durable subscriptions which request the queue manager manage the destination of publications.

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

non-durable-model-queue
Specify the name of the model queue.

Publish (PUBENBL)

Specifies whether messages can be published to the topic.

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

***ASPARENT**
Whether messages can be published to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***YES** Messages can be published to the topic.

***NO** Messages cannot be published to the topic.

Subscribe (SUBENBL)

Specifies whether applications are to be permitted to subscribe to this topic.

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

***ASPARENT**
Whether applications can subscribe to this topic is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***YES** Subscriptions can be made to this topic.

***NO** Applications cannot subscribe to this topic.

Default message priority (DFTPTY)

Specifies the default priority of messages published to the topic.

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

***ASPARENT**
The default priority is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

priority-value
Specify a value ranging from 0 through 9.

Default message persistence (DFTMSGPST)

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

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

***ASPARENT**
The default persistence is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

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

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

Default Put Response (DFTPUTRESP)

Specifies the type of response required for MQPUT and MQPUT1 calls when applications specify the MQPMO_RESPONSE_AS_Q_DEF option.

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

***ASPARENT**
The default response type is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***SYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are issued as if MQPMO_SYNC_RESPONSE had been specified instead. Fields in the MQMD and MQPMO are returned by the queue manager to the application.

***ASYNC**
Specifying this value ensures that the put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are always issued as if MQPMO_ASYNC_RESPONSE had been specified instead. Some fields in the MQMD and MQPMO are not returned by the queue manager to the application. An improvement in performance may be seen for messages put in a transaction or any non-persistent messages.

Wildcard behaviour (WILDCARD)

Specifies the behaviour of wildcard subscriptions with respect to this topic.

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

***PASSTHRU**
Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will receive publications made to this topic and to topic strings more specific than this topic.

***BLOCK**
Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will not receive publications made to this topic or to topic strings more specific than this topic.

Persistent message delivery (PMSGDLV)

Specifies the delivery mechanism for persistent messages published to this topic.

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

***ASPARENT**
The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***ALL** Persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

***ALLDUR**
Persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

***ALLAVAIL**
Persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

Non-persistent message delivery (NPMSGDLV)

Specifies the delivery mechanism for non-persistent messages published to this topic.

The possible values are:

***SYSDFTTOP**
The value of this attribute is taken from the system default topic.

***ASPARENT**
The delivery mechanism used is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***ALL** Non-persistent messages must be delivered to all subscribers, irrespective of durability for the MQPUT call to report success. If a delivery failure to any subscriber occurs, no other subscribers receive the message and the MQPUT call fails.

***ALLDUR**
Non-persistent messages must be delivered to all durable subscribers. Failure to deliver a persistent message to any non-durable subscribers does not return an error to the MQPUT call. If a delivery failure to a durable subscriber occurs, no subscribers receive the message and the MQPUT call fails.

***ALLAVAIL**
Non-persistent messages are delivered to all subscribers that can accept the message. Failure to deliver the message to any subscriber does not prevent other subscribers from receiving the message.

Examples

None

Error messages

Unknown

Convert MQ Data Type (CVTMQMDTA)

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

The Convert MQ Data Type (CVTMQMDTA) command produces a fragment of code to perform data conversion on data type structures, for use by the data-conversion exit program.

For information on how to use the data-conversion exit, see the WebSphere MQ Application Programming Guide.

Support is provided for the C programming language only.

Parameters

Keyword	Description	Choices	Notes
FROMFILE	Input file	*Qualified object name*	Required, Positional 1
	Qualifier 1: Input file	*Name*	
	Qualifier 2: Library	*Name*, ***LIBL** , *CURLIB	
FROMMBR	Member containing input	*Name*	Required, Positional 2
TOFILE	File to receive output	*Qualified object name*	Required, Positional 3
	Qualifier 1: File to receive output	*Name*	
	Qualifier 2: Library	*Name*, ***LIBL** , *CURLIB	
TOMBR	Member to receive output	*Name*, ***FROMMBR**	Optional, Positional 4
RPLTOMBR	Replace to member	***YES** , *NO	Optional, Positional 5

Input file (FROMFILE)

Specifies the qualified name of the file, in the form LIBRARY/FILE, that contains the data to convert.

The possible values are:

***LIBL** The library list is searched for the file name.

***CURLIB**
The current library is used.

from-library-name
Specify the name of the library to be used.

from-file-name
Specify the name of the file containing the data to convert.

Member containing input (FROMMBR)

Specifies the name of the member containing the data to be converted.

The possible values are:

from-member-name
Specifies the name of the member containing the data to convert.

File to receive output (TOFILE)

Specifies the qualified name of the file, in the form LIBRARY/FILE, that contains the converted data.

The possible values are:

***LIBL** The library list is searched for the file name.

***CURLIB**
The current library is used.

to-library-name
Specify the name of the library to be used.

to-file-name
Specify the name of the file to contain the converted data.

Member to receive output (TOMBR)

Specifies the name of the member containing the converted data.

The possible values are:

***FROMMBR**
The from-member name is used.

to-member-name
Specify the name of the member containing the converted data.

Replace to member (RPLTOMBR)

Specifies whether the converted data replaces the existing member.

The possible values are:

***YES** The converted data replaces the existing member.

***NO** The converted data does not replace the existing member.

Examples

None

Error messages

Unknown

Delete Message Queue Manager (DLTMQM)

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

The Delete Message Queue Manager (DLTMQM) command deletes the specified local queue manager.

Parameters

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

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

queue-manager-name
Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Examples

None

Error messages

Unknown

Delete MQ AuthInfo object (DLTMQMAUTI)

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

The Delete MQ AuthInfo object (DLTMQMAUTI) command deletes an existing MQ authentication information object.

Parameters

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

AuthInfo name (AINAME)

The name of the authentication information object to delete.

If an application has this open, the command fails.

The possible values are:

authentication-information-name
Specify the name of the authentication information object. The maximum string length is 48 characters.

Message Queue Manager name (MQMNAME)

The name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of an existing message queue manager. The maximum string length is 48 characters.

Examples

None

Error messages

Unknown

Delete MQ Pub/Sub Broker (DLTMQMBRK)

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

The delete WebSphere MQ broker command (DLTMQMBRK) is used to delete the broker. The broker must be stopped when this command is issued, and the queue manager must be running. If the broker is already started, you must issue ENDMQMBRK before issuing this command. To delete more than one broker in the hierarchy, it is essential that you stop (using the ENDMQMBRK command) and delete each broker one at a time. You should not attempt to stop all the brokers in the hierarchy that you want to delete first and then try to delete them.

Parameters

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

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

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

Examples

None

Error messages

Unknown

Delete MQ Channel (DLTMQMCHL)

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

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

Parameters

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

Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

channel-name
Specify the channel name.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Delete MQ Channel (CHLTYPE)

Specifies the type of the channel to delete.

The possible values are:

***NONCLT**
Any channel type, that is not a client-connection channel, that matches the channel name.

***SDR** Sender channel

***SVR** Server channel

***RCVR**
Receiver channel

***RQSTR**
Requester channel

***SVRCN**
Server-connection channel

***CLUSSDR**
Cluster-sender channel

***CLUSRCVR**
Cluster-receiver channel

***CLTCN**
Client-connection channel

Examples

None

Error messages

Unknown

Delete MQ Listener (DLTMQMLSR)

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

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

Parameters

Keyword	Description	Choices	Notes
LSRNAME	Listener name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Listener name (LSRNAME)

The name of the listener object to delete.

The possible values are:

listener-name
Specify the name of the listener definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

Delete MQ Namelist (DLTMQMNL)

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

The Delete MQ Namelist (DLTMQMNL) command deletes the specified namelist on the selected local queue manager.

Parameters

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

Namelist (NAMELIST)

The name of the namelist to delete.

namelist
Specify the name of the namelist. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used.

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

Examples

None

Error messages

Unknown

Delete MQ Process (DLTMQMPRC)

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

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

Parameters

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

Process name (PRCNAME)

The name of the process definition to delete. If an application has this process open, the command fails.

The possible values are:

process-name
Specify the name of the process definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

Delete MQ Queue (DLTMQMQ)

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

The Delete MQ Queue (DLTMQMQ) command deletes an MQ queue.

If the queue is a local queue, it must be empty for the command to succeed. CLRMQMQ can be used to clear all of the messages from a local queue.

The command fails if an application has:

- This queue open
- A queue that resolves to this queue open
- A queue open that resolves through this definition as a queue manager alias.

An application using the definition as a reply-to queue alias, however, does not cause this command to fail.

Parameters

Keyword	Description	Choices	Notes
QNAME	Queue name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Queue name (QNAME)

The name of the queue.

The possible values are:

queue-name
Specify the name of the queue.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Examples

None

Error messages

Unknown

Delete MQ Subscription (DLTMQMSUB)

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

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

Parameters

Keyword	Description	Choices	Notes
SUBID	Subscription identifier	*Character value*, ***NONE**	Optional, Positional 1
SUBNAME	Subscription name	*Character value*, ***NONE**	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3

Subscription identifier (SUBID)

The subscription identifier of the subscription to delete.

The possible values are:

subscription-name
Specify a maximum of 256 bytes for the subscription name.

Note: Subscription names of greater than 256 bytes can be speicifed using MQSC.

Subscription name (SUBNAME)

The name of the subscription to delete.

The possible values are:

subscription-name
Specify a maximum of 256 bytes for the subscription name.

Note: Subscription names of greater than 256 bytes can be speicifed using MQSC.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Examples

None

Error messages

Unknown

Delete MQ Service (DLTMQMSVC)

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

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

Parameters

Keyword	Description	Choices	Notes
SVCNAME	Service name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Service name (SVCNAME)

The name of the service object to delete.

The possible values are:

service-name
Specify the name of the service definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

Delete MQ Topic (DLTMQMTOP)

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

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

Parameters

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

Topic name (TOPNAME)

The name of the topic object to delete. If an application has this topic open, the command fails.

The possible values are:

topic-name
Specify the name of the topic object. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Examples

None

Error messages

Unknown

Disconnect MQ (DSCMQM)

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

The Disconnect Message Queue Manager (DSCMQM) command does not perform any function and is provided only for compatibility with previous releases of WebSphere MQ and MQSeries.

Parameters

None

Examples

None

Error messages

Unknown

Display Message Queue Manager (DSPMQM)

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

The Display Message Queue Manager (DSPMQM) command displays the attributes of the specified local queue manager.

Parameters

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

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

***** Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Examples

None

Error messages

Unknown

Display MQ Object Authority (DSPMQMAUT)

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

The Display MQ Authority (DSPMQMAUT) command shows, for the specified object, the current authorizations to the object. If a user ID is a member of more than one group, this command displays the combined authorizations of all of the groups.

- The 48-character MQ object name
- The MQ object type

- Authorizations for object, context and MQI calls

Parameters

Keyword	Description	Choices	Notes
OBJ	Object name	*Character value*	Required, Positional 1
OBJTYPE	Object type	*Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *MQM, *NMLIST, *PRC, *LSR, *SVC, *CHL, *CLTCN, *TOPIC	Required, Positional 2
USER	User name	*Name*, ***PUBLIC**	Optional, Positional 3
OUTPUT	Output	*, *PRINT	Optional, Positional 4
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 5
SRVCOMP	Service Component name	*Character value*, ***DFT**	Optional, Positional 6

Object name (OBJ)

Specifies the name of the MQ object for which the authorizations are displayed.

Object type (OBJTYPE)

Specifies the type of the object for which the authorizations are displayed.

***Q** All queue object types.

***ALSQ**
Alias queue.

***LCLQ**
Local queue.

***MDLQ**
Model queue.

***RMTQ**
Remote queue.

***AUTHINFO**
Authentication Information object.

***MQM**
Message Queue Manager.

***NMLIST**
Namelist object.

***PRC** Process definition.

***CHL** Channel object.

***CLTCN**
Client Connection Channel object.

***LSR** Listener object.

***SVC** Service object.

***TOPIC**
Topic object.

User name (USER)

Specifies the name of the user for whom authorities for the named object are displayed.

The possible values are:

***PUBLIC**
All users of the system.

user-profile-name
Specify the name of the user.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

*****
Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Service Component name (SRVCOMP)

Specifies the name of the installed authorization service in which to search for the authority to display.

The possible values are:

***DFT** All installed authorization components are searched for the specified object name, object type and user.

Authorization-service-component-name
The component name of the required authorization service as specified in the Queue Manager's qm.ini file.

Examples

None

Error messages

Unknown

Display MQ AuthInfo object (DSPMQMAUTI)

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

The Display MQ AuthInfo object (DSPMQMAUTI) command displays the attributes of an existing MQ authentication information object.

Parameters

Keyword	Description	Choices	Notes
AINAME	AuthInfo name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
OUTPUT	Output	*Character value*, *, *PRINT	Optional, Positional 3

AuthInfo name (AINAME)

The name of the authentication information object to display.

The possible values are:

authentication-information-name
Specify the name of the authentication information object. The maximum string length is 48 characters.

Message Queue Manager name (MQMNAME)

The name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of an existing message queue manager. The maximum string length is 48 characters.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

***** Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Examples

None

Error messages

Unknown

Display MQ Pub/Sub Broker (DSPMQMBRK)

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

The Display WebSphere MQ broker (DSPMQMBRK) command is used to display the current status of a broker.

Parameters

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

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

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

Examples

None

Error messages

Unknown

Display MQ Channel (DSPMQMCHL)

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

The Display MQ Channel (DSPMQMCHL) command displays the attributes of an existing MQ channel definition.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*	Required, Positional 1
OUTPUT	Output	*, *PRINT	Optional, Positional 2

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, ***NONCLT**, *CLTCN	Optional, Positional 4

Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

channel-name
Specify the channel name.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

***** Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Channel type (CHLTYPE)

Specifies the type of the channel to be displayed.

The possible values are:

***NONCLT**
Any channel type, that is not a client-connection channel, that matches the channel name.

***SDR** Sender channel

***SVR** Server channel

***RCVR**
Receiver channel

***RQSTR**
Requester channel

***SVRCN**
Server-connection channel

***CLUSSDR**
Cluster-sender channel

***CLUSRCVR**
Cluster-receiver channel

***CLTCN**
Client-connection channel

Examples

None

Error messages

Unknown

Display MQ Command Server (DSPMQMCSVR)

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

The Display MQ Command Server (DSPMQMCSVR) command displays the status of the MQ command server.

The status of the command server can be one of the following:

Enabled
Available to process messages

Disabled
Not available to process messages

Starting
STRMQMCSVR command in progress

Stopping
ENDMQMCSVR command in progress

Stopped
ENDMQMCSVR command completed

Running
Processing a message

Waiting
Waiting for a message

Parameters

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

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Examples

None

Error messages

Unknown

Display MQ Listener (DSPMQMLSR)

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

The Display MQ Listener object (DSPMQMLSR) command displays the attributes of an existing MQ listener object.

Parameters

Keyword	Description	Choices	Notes
LSRNAME	Listener name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
OUTPUT	Output	*, *PRINT	Optional, Positional 3

Listener name (LSRNAME)

The name of the listener object to display.

The possible values are:

listener-name
Specify the name of the listener definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

* Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Examples

None

Error messages

Unknown

Display MQ Namelist (DSPMQMNL)

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

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

Parameters

Keyword	Description	Choices	Notes
NAMELIST	Namelist	*Character value*	Required, Positional 1
OUTPUT	Output	*, *PRINT	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3

Namelist (NAMELIST)

The name of the namelist to be displayed.

namelist
Specify the name of the namelist. The maximum length of the string is 48 bytes.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

***** Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used.

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

Examples

None

Error messages

Unknown

Display MQ Object Names (DSPMQMOBJN)

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

The Display MQ Object Names (DSPMQMOBJN) command is used to provide the name, type, and fully-qualified file name for a specified MQ object.

Parameters

Keyword	Description	Choices	Notes
OBJ	Object name	*Character value*, *ALL	Required, Positional 1

Keyword	Description	Choices	Notes
OBJTYPE	Object type	***ALLMQM**, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *CTLG, *CHL, *CLTCN, *SVC, *MQM, *NMLIST, *PRC, *LSR, *TOPIC	Optional, Positional 2
OUTPUT	Output	*, *PRINT	Optional, Positional 3
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 4

Object name (OBJ)

Specifies the name of the objects for which the corresponding name, type and file name to display. It is a 48-character MQ object or generic object name.

The possible values are:

***ALL** All objects of the specified type (OBJTYPE) are displayed.

generic-object-name
Specify the generic name of the objects. A generic name is a character string followed by an asterisk (*). For example, ABC*. It selects all objects having names that start with the selected character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

object-name
The name of an object for which the corresponding name and type is to be displayed.

Object type (OBJTYPE)

Specifies the type of the objects to be displayed.

The possible values are:

***ALLMQM**
All MQ Objects with names specified by OBJ.

***Q** All MQ queues with names specified by OBJ.

***ALSQ**
All MQ alias queues with names specified by OBJ.

***LCLQ**
All MQ local queues with names specified by OBJ.

***MDLQ**
All MQ model queues with names specified by OBJ.

***RMTQ**
All MQ remote queues with names specified by OBJ.

***AUTHINFO**
All MQ authentication information objects with names specified by OBJ.

***CHL** All MQ channel objects with names specified by OBJ.

***CLTCN**
All MQ client connection channel objects with names specified by OBJ.

***SVC** All MQ service objects with names specified by OBJ.

***LSR** All MQ listener objects with names specified by OBJ.

***CTLG**
The MQ queue manager catalog object with name specified by OBJ. This has the same name as the queue manager object.

***MQM**
The Message Queue Manager object with name specified by OBJ.

***NMLIST**
All MQ namelists with names specified by OBJ.

***PRC** All MQ process definitions with names specified by OBJ.

***LOBJ** All MQ listener objects with names specified by OBJ.

***TOPIC**
All MQ topic objects with names specified by OBJ.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

***** Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Message Queue Manager name (MQMNAME)

Specifies the name of the MQ queue manager for which object information is to displayed.

The possible values are:

***DFT** The default queue manager.

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

Examples

None

Error messages

Unknown

Display MQ Process (DSPMQMPRC)

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

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

Parameters

Keyword	Description	Choices	Notes
PRCNAME	Process name	*Character value*	Required, Positional 1
OUTPUT	Output	*, *PRINT	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3

Process name (PRCNAME)

The name of the process definition to be displayed.

The possible values are:

process-name
Specify the name of the process definition. The maximum length of the string is 48 bytes.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

***** Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

Display MQ Queue (DSPMQMQ)

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

The Display MQ Queue (DSPMQMQ) command displays the attributes of an existing MQ queue definition.

Parameters

Keyword	Description	Choices	Notes
QNAME	Queue name	*Character value*	Required, Positional 1

Keyword	Description	Choices	Notes
OUTPUT	Output	*, *PRINT	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3

Queue name (QNAME)

The name of the queue.

The possible values are:

queue-name
Specify the name of the queue.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting work station, or printed with the job's spooled output.

The possible values are:

*
Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Examples

None

Error messages

Unknown

Display MQ Route Information (DSPMQMRTE)

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

The DSPMQMRTE command generates a trace route message based on user specified parameters and puts it to a specified queue. One or more reports about the route the message takes to its final destination might be generated, as well as a reply. These will be got from a specified reply queue and the information contained within them will be written to the job's spooled output when it is received.

Parameters

Keyword	Description	Choices	Notes
QNAME	Target queue	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
CRRLID	Correlation Identifier	*Character value*, ***NONE**	Optional, Positional 3
MSGPST	Message Persistence	*YES, ***NO**, *QUEUE	Optional, Positional 4
MSGPRTY	Message Priority	0-9, ***QUEUE**	Optional, Positional 5
OPTION	Report Option	Single values: ***DFT**, *NONE Other values (up to 6 repetitions): *ACTIVITY, *COA, *COD, *DISCARD, *EXCEPTION, *EXPIRATION	Optional, Positional 6
RPLYQ	Reply Queue	*Character value*, ***DFT**	Optional, Positional 7
RPLYMQM	Reply Queue Manager	*Character value*, ***DFT**	Optional, Positional 8
EXPIRY	Message Expiry	0-999999999, ***DFT**	Optional, Positional 9
EXPRPT	Pass Expiry	***YES**, *NO	Optional, Positional 10
RTEINF	Route Accumulation	*YES, ***NO**	Optional, Positional 11
RPLYMSG	Reply Message	*YES, ***NO**	Optional, Positional 12
DLVRMSG	Deliver Message	*YES, ***NO**	Optional, Positional 13
FWDMSG	Forward Message	***SUPPORT**, *ALL	Optional, Positional 14

Keyword	Description	Choices	Notes
MAXACTS	Maximum Activities	1-999999999, ***NOMAX**	Optional, Positional 15
DETAIL	Route Detail	*LOW, ***MEDIUM**, *HIGH	Optional, Positional 16
BROWSE	Browse Only	*YES, ***NO**	Optional, Positional 17
DSPMSG	Display Message	***YES**, *NO	Optional, Positional 18
TGTMQM	Target Queue Manager	*Character value*, ***DFT**	Optional, Positional 19
DSPINF	Display Information	Single values: ***ALL**, *SUMMARY, *NONE Other values (up to 6 repetitions): *ACTGRP, *ID, *MSGGRP, *MSGDELTA, *OPGRP, *TRGRP	Optional, Positional 20
WAIT	Wait Time	0-999999999, ***DFT**	Optional, Positional 21
BIND	Bind Option	***OPEN**, *NOTFIXED	Optional, Positional 22

Target queue (QNAME)

Specifies the name of the target queue of the trace route message or, if displaying previously gathered information, the name of the queue storing the information.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Correlation Identifier (CRRLID)

Specifies the CorrelId to use when retrieving previously gathered information. The format of the 24 byte CorrelId is a 48 character hexadecimal string. You must supply a CorrelId if you are retrieving previously gathered information, rather than generating a trace route message.

The possible values are:

***NONE**
No CorrelId is supplied.

correlation-identifier
The 48 character hexadecimal string representing the 24 byte CorrelId.

Message Persistence (MSGPST)

Specifies the persistence of the trace route message.

The possible values are:

***NO** The message will be put with MQPER_NOT_PERSISTENT.

***YES** The message will be put with MQPER_PERSISTENT.

***QUEUE**
The message will be put with MQPER_PERSISTENCE_AS_Q_DEF.

Message Priority (MSGPRTY)

Specifies the priority of the trace route message.

The possible values are:

***QUEUE**
The message will be put with MQPRI_PRIORITY_AS_Q_DEF.

message-priority
The priority of the message ranging 0 through 9.

Report Option (OPTION)

Specifies the report options of the trace route message. Reports generated on a non trace route enabled queue manager can potentially remain in the network undelivered, which is why most report options are disabled by default. By requesting full data to be returned, it allows the trace route information contained in the message to be returned in the result of a problem.

The possible values are:

***DFT** Turns on MQRO_ACTIVITY and MQRO_DISCARD_MSG.

***NONE**
No report options are set.

***ACTIVITY**
Turns on MQRO_ACTIVITY.

***COA** Turns on MQRO_COA_WITH_FULL_DATA.

***COD** Turns on MQRO_COD_WITH_FULL_DATA.

***DISCARD**
Turns on MQRO_DISCARD_MSG.

***EXCEPTION**
Turns on MQRO_EXCEPTION_WITH_FULL_DATA.

***EXPIRATION**
Turns on MQRO_EXPIRATION_WITH_FULL_DATA.

Reply Queue (RPLYQ)

Specifies the name of the reply queue to which the reply and all report messages should be sent. This must exist on the local queue manager unless the RPLYMQM parameter is also specified. The reply queue should not be a temporary queue if the trace route message is to be persistent.

The possible values are:

***DFT** The SYSTEM.DEFAULT.MODEL.QUEUE is used and the reply queue is by default a temporary dynamic queue.

reply-queue
The name of the reply queue to use.

Reply Queue Manager (RPLYMQM)

Specifies the queue manager to which replies are sent.

The possible values are:

***DFT** Replies are sent to the local queue manager.

reply-queue-manager
The name of the reply to queue manager.

Message Expiry (EXPIRY)

Specifies the Expiry time, in seconds, of the trace route message.

The possible values are:

***DFT** The default expiry time of 60 seconds is used.

expiry-time
The expiry time of the message ranging from 0 through 999999999.

Pass Expiry (EXPRPT)

Specifies whether the expiry of the trace route message is passed to reports or the reply message. This effectively turns MQRO_PASS_DISCARD_AND_EXPIRY on and off. This allows users to keep the reports indefinitely if required.

The possible values are:

***YES** Expiry is passed to reports or the reply message.

***NO** Expiry is not passed to reports or the reply message.

Route Accumulation (RTEINF)

Specifies that the route information is accumulated within the trace route message as it flows through the queue manager network.

The possible values are:

***NO** No information is accumulated within the trace route message.

***YES** Information is accumulated within the trace route message.

Reply Message (RPLYMSG)

Requests that a reply message containing all accumulated information is returned to the reply to queue when the trace route message reaches its final destination (if this is permitted by the queue manager hosting the final destination queue).

The possible values are:

***NO** No reply message is returned.

***YES** A reply message is returned to the the reply to queue.

Deliver Message (DLVRMSG)

Specifies whether the trace route message is delivered to getting applications if the message successfully arrives at the destination queue.

The possible values are:

***NO** If the trace route message successfullly arrives at the target queue it is not delivered to getting applications.

***YES** The trace route message is delivered to a getting application if the message successfully arrives at the target queue. Specifying this option effectively gives permission for the message to arrive on a queue manager, whether it supports trace route or not.

Forward Message (FWDMSG)

Specifies whether the trace route message is forwarded to the next queue manager in the route.

The possible values are:

***SUPPORT**
The trace route message is forwarded only to queue managers that can ensure that the delivery option is honoured.

***ALL** The trace route message is forwarded on without any regard given to the next queue manager in the route. This option can be used to force a non-trace route enabled queue manager to accept trace route messages, even when they cannot process them in line with the delivery option.

Maximum Activities (MAXACTS)

Specifies the maximum number of activities that can take place on the trace route message before it is discarded.

The possible values are:

***NOMAX**
No maximum number of activities are specified.

maximum-activities
The maximum numer of activities ranging from 1 through 999999999.

Route Detail (DETAIL)

Specifies how much detail about the route is requested.

The possible values are:

***LOW** At this level of detail no information about queue manager activities is requested. This gives a very high level view of what user activity has taken place on the message.

***MEDIUM**
Low detail information, as well as information on the movements of the message within the queue manager is requested. This includes the work of the MCA.

***HIGH**
Low and medium detail, as well as more detailed information about the route the message took is requested. For example, in clustering this might include detail about why the route was chosen.

Browse Only (BROWSE)

Specifies whether messages returned are browsed only. This means that the information remains on the queue for future display operations.

The possible values are:

***NO** Messages returned are not browse only.

***YES** Messages returned are browse only.

Display Message (DSPMSG)

Specifies whether when a trace route message is generated the information returned is displayed.

The possible values are:

***YES** The returned information is displayed.

***NO** The returned information is not displayed. This allows DSPMQMRTE to exit as soon as the trace route message has been put to the target queue. On exit, a 48 character hexadecimal string is output, which is the MsgId on the trace route message that was generated and can be used as the CRRLID supplied to a subsequent DSPMQMRTE call.

Target Queue Manager (TGTMQM)

Specifies the target queue manager for the trace route message.

The possible values are:

***DFT** No target queue manager is specified. Either the destination queue is a local queue, or there is a local definition of the queue.

target-queue-manager
The target queue manager for the trace route message.

Display Information (DSPINF)

Specifies how much of the information gathered should be displayed.

The possible values are:

***ALL** All available information is displayed.

***SUMMARY**
Displays only the queues which the message was routed through.

***NONE**
None of the available information will be displayed.

***ACTGRP**
All non-group parameters in the Activity groups will be displayed.

***ID** Values with parameters identifiers MQBACF_MSG_ID or

MQBACF_CORREL_ID are always displayed. This overrides *MSGDELTA which normally prevents certain values in the Message groups from being displayed.

***MSGGRP**
All non-group parameters in the Message groups are displayed.

***MSGDELTA**
Like *MSGGRP, except that information in the Message groups is only displayed where it has changed since the last operation took place.

***OPGRP**
All non-group parameters in the Operation groups are displayed.

***TRGRP**
All parameters in the TraceRoute groups are displayed.

Wait Time (WAIT)

Specifies how long, in seconds, that DSPMQMRTE should wait before assuming that all a reply message or all the reports (depending on the options specified) that were generated en route that can be delivered to the reply queue have now done so.

The possible values are:

***DFT** DSPMQMRTE waits for 60 seconds longer than the Expiry time of the trace route message.

wait-time
The time that DSPMQMRTE should wait.

Bind Option (BIND)

Specifies whether the target queue is bound to a specific destination.

The possible values are:

***OPEN**
The target queue is bound to a specific destination. The queue is opened with option MQOO_BIND_ON_OPEN.

***NOTFIXED**
The target queue is not bound to a specific destination. Typically this parameter is used when the trace route message is to be put across a cluster. The queue is opened with option MQOO_BIND_NOT_FIXED.

Examples

None

Error messages

Unknown

Display Queue Manager Status (DSPMQMSTS)

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

The Display Message Queue Manager Status (DSPMQMSTS) command displays the status attributes of the specified local queue manager.

Parameters

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

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

* Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Examples

None

Error messages

Unknown

Display MQ Subscription (DSPMQMSUB)

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

The Display MQ Subscription (DSPMQMSUB) command displays the attributes of an existing MQ subscription.

Parameters

Keyword	Description	Choices	Notes
SUBID	Subscription identifier	*Character value*, ***NONE**	Optional, Positional 1
SUBNAME	Subscription name	*Character value*, ***NONE**	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
OUTPUT	Output	*, *PRINT	Optional, Positional 4

Subscription identifier (SUBID)

The subscription identifier of the subscription to be displayed.

The possible values are:

subscription-name
Specify a maximum of 256 bytes for the subscription name.

Note: Subscription names of greater than 256 bytes can be speicifed using MQSC.

Subscription name (SUBNAME)

The name of the subscription to be displayed.

The possible values are:

subscription-name
Specify a maximum of 256 bytes for the subscription name.

Note: Subscription names of greater than 256 bytes can be speicifed using MQSC.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

***** Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Examples

None

Error messages

Unknown

Display MQ Service (DSPMQMSVC)

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

The Display MQ Service object (DSPMQMSVC) command displays the attributes of an existing MQ service object.

Parameters

Keyword	Description	Choices	Notes
SVCNAME	Service name	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Key, Positional 2
OUTPUT	Output	*, *PRINT	Optional, Positional 3

Service name (SVCNAME)

The name of the service object to display.

The possible values are:

service-name
Specify the name of the service definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting work station or printed with the job's spooled output.

The possible values are:

*
: Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
: The output is printed with the job's spooled output.

Examples

None

Error messages

Unknown

Display MQ Topic (DSPMQMTOP)

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

The Display MQ Topic (DSPMQMTOP) command displays the attributes of an existing MQ topic object.

Parameters

Keyword	Description	Choices	Notes
TOPNAME	Topic name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
OUTPUT	Output	*, *PRINT	Optional, Positional 3

Topic name (TOPNAME)

The name of the topic object to be displayed.

The possible values are:

topic-name
Specify the name of the topic object. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation or printed with the job's spooled output.

The possible values are:

***** Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Examples

None

Error messages

Unknown

Display MQ Version (DSPMQMVER)

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

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

Parameters

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

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

The possible values are:

* Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
The output is printed with the job's spooled output.

Examples

None

Error messages

Unknown

End Message Queue Manager (ENDMQM)

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

The End Message Queue Manager (ENDMQM) command ends the specified local message queue manager or all queue managers. The attributes of the message queue managers are not affected and it can be restarted using the Start Message Queue Manager (STRMQM) command.

You can also use this command to fully quiesce all application programs connected to the queue manager or all queue managers.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
OPTION	Option	***CNTRLD** , *IMMED, *WAIT, *PREEMPT	Optional, Positional 2
ENDCCTJOB	End connected jobs	***NO** , *YES	Optional, Positional 3
RCDMQMIMG	Record MQ Object Image	*NO, ***YES**	Optional, Positional 4
TIMEOUT	Timeout interval (seconds)	0-3600, **30**	Optional, Positional 5

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

***ALL** All queue managers are ended.

Option (OPTION)

Specifies whether processes that are connected to the queue manager are allowed to complete.

The possible values are:

***CNTRLD**
Allow programs currently being processed to complete. An MQCONN call (or an MQOPEN or MQPUT1, which perform an implicit connection) fails. If ENDCCTJOB(*YES) is specified, a controlled shutdown of the queue manager is attempted ten times. If the queue manager shuts down successfully, it is followed by immediate termination of the processes that are still connected to it.

***IMMED**
End the queue manager immediately. All current MQI calls complete, but subsequent requests for MQI calls fail. Incomplete units of work are rolled back when the queue manager is next started. If ENDCCTJOB(*YES) is specified, a controlled shutdown of the queue manager is followed if necessary, after an interval of TIMEOUT seconds, by an immediate

shutdown of the queue manager. This is followed by immediate termination of processes connected to it.

***WAIT**
End the queue manager in the same way as the *CNTRLD option. However, control is returned only after the queue manager has stopped. This option is not allowed with MQMNAME(*ALL). If ENDCCTJOB(*YES) is specified, a single controlled shutdown of the queue manager is issued, which waits for all processes to disconnect. When this completes it is followed by the actions described in the ENDCCTJOB parameter.

***PREEMPT**
Use this type of shutdown only in exceptional circumstances The queue manager stops without waiting for applications to disconnect or for MQI calls to complete. This can give unpredictable results for WebSphere MQ applications. All processes in the queue manager that fail to stop are ended 30 seconds after the command is issued. This option is not allowed with ENDCCTJOB(*YES).

End connected jobs (ENDCCTJOB)

Specifies whether all processes connected to the queue manager are forcibly terminated.

The possible values are:

***NO** The queue manager or queue managers are ended but no further action is taken.

***YES** The following steps are taken for each queue manager to be ended:

- If the queue manager is running and RCDMQMIMG(*YES) has been specified, media images for all objects defined for the queue manager are recorded.
- The queue manager is ended in the appropriate manner (*CNTRLD, *WAIT, or *IMMED).
- All shared memory and semaphores used by the queue manager are deleted irrespective of whether applications have disconnected from the queue manager. Applications that have not disconnected from a a shared memory resource when this option is specified receive a return code of MQRC_CONNECTION_BROKEN (2009) the next time an MQI call is issued with an existing connection handle.

Record MQ Object Image (RCDMQMIMG)

Specifies whether media images are recorded for a queue manager.

The possible values are:

***YES** If the queue manager is running, media images for all queue manager objects are recorded.

***NO** Media images of queue manager objects are not recorded as part of the quiesce.

Timeout interval (seconds) (TIMEOUT)

Specifies the time interval in seconds between the controlled and immediate shutdowns of the queue manager when *IMMED is specified. It also determines the number of seconds between attempts to shut down the queue manager when *CNTRLD is specified.

The possible values are:

30 The default value is 30 seconds.

timeout-interval
Specify a value in seconds, in the range 0 through 3600.

Examples

None

Error messages

Unknown

End MQ Pub/Sub Broker (ENDMQMBRK)

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

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

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*	Required, Positional 1
OPTION	Option	***CNTRLD** , *IMMED	Optional, Positional 2

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

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

Option (OPTION)

Specifies how the broker is ended.

The possible values are:

***CNTRLD**
Allows the broker to complete processing for any message that it has already started.

***IMMED**
Ends the broker immediately. The broker does not attempt any further gets or puts, and backs out any in-flight units-of-work. This might mean that a nonpersistent input message is published only to a subset of subscribers, or lost, depending on the broker configuration parameters.

Examples

None

Error messages

Unknown

End MQ Channel (ENDMQMCHL)

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

The End MQ Channel (ENDMQMCHL) command closes an MQ channel, and the channel is no longer enabled for automatic restarts.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*	Required, Positional 1

Keyword	Description	Choices	Notes
OPTION	Option	***CNTRLD**, *IMMED, *ABNORMAL	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
STATUS	Channel status	***STOPPED**, *INACTIVE	Optional, Positional 4
CONNAME	Connection name	*Character value*, ***NONE**	Optional, Positional 5
RQMNAME	Remote queue manager	*Character value*, ***NONE**	Optional, Positional 6

Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

channel-name
Specify the channel name.

Option (OPTION)

Specifies whether processing for the current batch of messages is allowed to finish in a controlled manner.

The possible values are:

***CNTRLD**
Allows processing of the current batch of messages to complete. No new batch is allowed to start.

***IMMED**
Ends processing of the current batch of messages immediately. This is likely to result in 'in-doubt' situations.

***ABNORMAL**
Ends processing of the current batch of messages immediately and terminates the channel thread or job. This is likely to result in 'in-doubt' situations.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Channel status (STATUS)

Specifies the required status of the channel after succesful completion of the command.

The possible values are:

***STOPPED**
The channel status is set to STOPPED.

***INACTIVE**
The channel status is set to INACTIVE.

Connection name (CONNAME)

Specifies the connection name of the channel instance that you want to end.

Remote queue manager (RQMNAME)

Specifies the name of the remote queue manager of the channel instance that you want to end.

Examples

None

Error messages

Unknown

End Queue Manager Connection (ENDMQMCONN)

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

The End MQ Connections (ENDMQMCONN) command allows you to end a connection to the queue manager.

Parameters

Keyword	Description	Choices	Notes
CONN	Connection Identifier	*Character value*	Required, Key, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Connection Identifier (CONN)

The connection identifier to end.

The connection identifier is a 16 character hex string.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

End MQ Command Server (ENDMQMCSVR)

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

The End MQ Command Server (ENDMQMCSVR) command stops the MQ command server for the specified local queue manager.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*	Required, Positional 1
OPTION	Option	***CNTRLD**, *IMMED	Optional, Positional 2

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

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

Option (OPTION)

Specifies whether or not the command message currently being processed is allowed to complete.

The possible values are:

***CNTRLD**
Allows the command server to complete processing any command message that it has already started. No new message is read from the queue.

***IMMED**
Ends the command server immediately. Any action associated with a command message currently being processed might not be completed.

Examples

None

Error messages

Unknown

End MQ Listeners (ENDMQMLSR)

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

The End MQ Listener (ENDMQMLSR) command ends an MQ TCP/IP listener.

This command is valid only for TCP/IP transmission protocols.

Either a listener object or specific port can be specified.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
PORT	Port number	1-65535, *ALL	Optional, Positional 2
OPTION	Option	***CNTRLD**, *WAIT, *FORCE	Optional, Positional 3
LSRNAME	Listener name	*Character value*, ***NONE**	Optional, Positional 4

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Port number (PORT)

The port number to be used by the listener.

The possible values are:

***SAME**
The attribute is unchanged.

port-number
The port number to be used.

Option (OPTION)

Specifies the action taken after processes to end the listeners have been started.

***CNTRLD**
Processes are started to end all the listeners for the specified queue manager and control is returned before the listeners actually end.

***WAIT**
End the listeners for the specified queue manager in the same way as the *CNTRLD option. However, control is returned only after all the listeners have ended.

Listener name (LSRNAME)

The name of the MQ listener object to end.

The possible values are:

***NONE**
No listener object is specified.

listener-name
Specify the name of the listener definition. The maximum length of the string is 48 bytes.

Examples

None

Error messages

Unknown

End MQ Service (ENDMQMSVC)

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

The End MQ Service (ENDMQMSVC) command ends an MQ service.

Parameters

Keyword	Description	Choices	Notes
SVCNAME	Service name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Service name (SVCNAME)

The name of the MQ service object to end.

The possible values are:

***NONE**
No service object is specified.

service-name
Specify the name of the service definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

Grant MQ Object Authority (GRTMQMAUT)

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

The Grant MQ Authority (GRTMQMAUT) command is used to grant specific authority for the MQ objects named in the command to another user or group of users.

Authority can be given to:

- Named users.
- Users (*PUBLIC) who do not have authority specifically given to them.
- Groups of users who do not have any authority to the object.

The GRTMQMAUT command can be used by anyone in the QMQMADM group, that is, anyone whose user profile specifies QMQMADM as a primary or supplemental group profile.

Parameters

Keyword	Description	Choices	Notes
OBJ	Object name	*Character value*	Required, Positional 1
OBJTYPE	Object type	*ALL, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *MQM, *NMLIST, *PRC, *LSR, *SVC, *CHL, *CLTCN, *TOPIC	Required, Positional 2
USER	User names	Single values: *PUBLIC Other values (up to 50 repetitions): *Name*	Required, Positional 3
AUT	Authority	Values (up to 21 repetitions): *ALTUSR, *BROWSE, *CONNECT, *GET, *INQ, *PUT, *SET, *PUB, *SUB, *RESUME, *PASSALL, *PASSID, *SETALL, *SETID, *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP, *ALL, *ALLADM, *ALLMQI, *NONE, *CTRL, *CTRLX	Required, Positional 4

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 5
SRVCOMP	Service Component name	*Character value*, ***DFT**	Optional, Positional 6

Object name (OBJ)

Specifies the name of the objects for which specific authorities are granted.

The possible values are:

***ALL** All objects of the type specified by the value of the OBJTYPE parameter at the time the command is issued. *ALL cannot represent a generic profile.

object-name
Specify the name of an MQ object for which specific authority is given to one or more users.

generic profile
Specify the generic profile of the objects to be selected. A generic profile is a character string containing one or more generic characters anywhere in the string. This profile is used to match the object name of the object under consideration at the time of use. The generic characters are (?), (*) and (**).

? matches a single character in an object name.

* matches any string contained within a qualifier, where a qualifier is the string between fullstops (.). For example ABC* matches ABCDEF but not ABCDEF.XYZ.

** matches one or more qualifiers. For example ABC.**.XYZ matches ABC.DEF.XYZ and ABC.DEF.GHI.XYZ, ** can only appear once in a generic profile.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

Object type (OBJTYPE)

Specifies the type of the objects for which specific authorities are granted.

***ALL** All MQ object types.

***Q** All queue object types.

***ALSQ**
Alias queue.

***LCLQ**
Local queue.

***MDLQ**
Model queue.

***RMTQ**
Remote queue.

***AUTHINFO**
Authentication Information object.

***MQM**
Message Queue Manager.

***NMLIST**
Namelist object.

***PRC** Process definition.

***CHL** Channel object.

***CLTCN**
Client Connection Channel object.

***LSR** Listener object.

***SVC** Service object.

***TOPIC**
Topic object.

User names (USER)

Specifies the name or names of users to whom authorities for the named object are being given. If user names are specified, the authorities are given specifically to those users. Authority given by this command can be revoked specifically by the Revoke MQ Authority (RVKMQMAUT) command.

***PUBLIC**
All users of the system.

user-profile-name
Specify the names of one or more users who are to be granted specific authority for the object. You can specify up to 50 user profile names.

Authority (AUT)

Specifies the authority being given to the named users. Values for AUT can be specified as a list of specific and general authorities in any order, where the general authorities can be:

*NONE, which creates a profile for the user with no authority to the specified object, or leaves the authority unchanged if a profile already exists.

*ALL, which confers all authorities to the specified users.

*ALLADM, which confers all of *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP, *CTRL and *CTRLX.

*ALLMQI, which confers all of *ALTUSR, *BROWSE, *CONNECT, *GET, *INQ, *PUT, *SET, *PUB, *SUB and *RESUME.

Authorizations for different object types

***ALL** All authorizations. Applies to all objects.

***ADMCHG**
Change an object. Applies to all objects.

***ADMCLR**
Clear a queue. Applies to queues only.

***ADMCRT**
Create an object. Applies to all objects.

***ADMDLT**
Delete an object. Applies to all objects.

***ADMDSP**
Display the attributes of an object. Applies to all objects.

***ALLADM**
Perform administration operations on an object. Applies to all objects.

***ALLMQI**
Use all MQI calls applicable to an object. Applies to all objects.

***ALTUSR**
Allow another user's authority to be used for MQOPEN and MQPUT1 calls. Applies to queue manager objects only.

***BROWSE**
Retrieve a message from a queue by issuing an MQGET call with the BROWSE option. Applies to queue objects only.

***CONNECT**
Connect the application to a queue manager by issuing an MQCONN call. Applies to queue manager objects only.

***CTRL**
Control startup and shutdown of channels, listeners and services.

***CTRLX**
Reset sequence number and resolve indoubt channels.

***GET** Retrieve a message from a queue using an MGET call. Applies to queue objects only.

***INQ** Make an inquiry on an object using an MQINQ call. Applies to all objects.

***PASSALL**
Pass all context on a queue. Applies to queue objects only.

***PASSID**
Pass identity context on a queue. Applies to queue objects only.

***PUT** Put a message on a queue using an MQPUT call. Applies to queue objects only.

***SET** Set the attributes of an object using an MQSET call. Applies to queue, queue manager, and process objects only.

***SETALL**
Set all context on an object. Applies to queue and queue manager objects only.

***SETID**
Set identity context on an object. Applies to queue and queue manager objects only.

Authorizations for MQI calls

***ALTUSR**
Allow another user's authority to be used for MQOPEN and MQPUT1 calls.

***BROWSE**
Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

***CONNECT**
Connect the application to the specified queue manager by issuing an MQCONN call.

***GET** Retrieve a message from a queue by issuing an MQGET call.

***INQ** Make an inquiry on a specific queue by issuing an MQINQ call.

***PUT** Put a message on a specific queue by issuing an MQPUT call.

***SET** Set attributes on a queue from the MQI by issuing an MQSET call.

***PUB** Open a topic to publish a message using the MQPUT call.

***SUB** Create, Alter or Resume a subscription to a topic using the MQSUB call.

***RESUME**
Resume a subscription using the MQSUB call.

If you open a queue for multiple options, you must be authorized for each of them.

Authorizations for context

***PASSALL**
Pass all context on the specified queue. All the context fields are copied from the original request.

***PASSID**
Pass identity context on the specified queue. The identity context is the same as that of the request.

***SETALL**
Set all context on the specified queue. This is used by special system utilities.

***SETID**
Set identity context on the specified queue. This is used by special system utilities.

Authorizations for MQSC and PCF commands

***ADMCHG**
Change the attributes of the specified object.

***ADMCLR**
Clear the specified queue (PCF Clear queue command only).

***ADMCRT**
Create objects of the specified type.

***ADMDLT**
Delete the specified object.

***ADMDSP**
Display the attributes of the specified object.

***CTRL**
Control startup and shutdown of channels, listeners and services.

***CTRLX**
Reset sequence number and resolve indoubt channels.

Authorizations for generic operations

***ALL** Use all operations applicable to the object.

***ALLADM**
Perform all administration operations applicable to the object.

***ALLMQI**
Use all MQI calls applicable to the object.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Service Component name (SRVCOMP)

Specifies the name of the installed authorization service to which the authorizations apply.

The possible values are:

***DFT** Use the first installed authorization component.

Authorization-service-component-name
The component name of the required authorization service as specified in the Queue Manager's qm.ini file.

Examples

None

Error messages

Unknown

Ping MQ Channel (PNGMQMCHL)

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

The Ping MQ Channel (PNGMQMCHL) command tests a channel by sending data as a special message, to the remote message queue manager and checks that the data is returned. This command is successful only from the sending end of an inactive channel, and the data used is generated by the local message queue manager.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
DATACNT	Data count	16-32768, **64**	Optional, Positional 3
CNT	Count	1-16, **1**	Optional, Positional 4

Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

channel-name
Specify the channel name.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Data count (DATACNT)

Specifies the length of the data in bytes. The actual number of bytes might be less than the amount requested depending on the operating system and communication protocol being used.

The possible values are:

64 The default value is 64 bytes.

data-count Specify a value ranging from 16 through 32768.

Count (CNT)

Specifies the number of times that the channel is to be pinged.

The possible values are:

1 The channel is pinged once.

ping-count Specify a value ranging from 1 through 16.

Examples

None

Error messages

Unknown

Record MQ Object Image (RCDMQMIMG)

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

The Record MQ Object Image (RCDMQMIMG) command is used to provide a marker for the selected set of MQ objects, so that the Recreate MQM Object (RCRMQMOBJ) command can recover this set of objects from journal data recorded subsequently.

This command is intended to enable journal receivers, detached prior to the current date, to be disconnected. On successful completion of this command those

journals are no longer required to be present for a Recreate MQ Object (RCRMQMOBJ) command on this set of MQM Objects to succeed.

Parameters

Keyword	Description	Choices	Notes
OBJ	Object name	*Character value*, *ALL	Required, Positional 1
OBJTYPE	Object type	*ALL, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *CTLG, *MQM, *NMLIST, *PRC, *CHL, *CLTCN, *LSR, *SVC, *SYNCFILE, *TOPIC	Required, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
DSPJRNDTA	Display Journal Receiver Data	*YES, ***NO**	Optional, Positional 4

Object name (OBJ)

Specifies the name of the objects that should be recorded. This is a 48-character MQ object or generic object name.

The possible values are:

***ALL** All MQ objects of the specified type (OBJTYPE) are recorded.

generic-object-name
Specify the generic name of the objects to be recorded. A generic name is a character string followed by an asterisk (*). For example, ABC*. It selects all objects that have names which start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

object-name
The name of an MQ object to be recorded.

Object type (OBJTYPE)

Specifies the type of the objects to be recreated.

The possible values are:

***ALL** Specifies all MQ object types.

***Q** Specifies MQ queue objects with names specified by OBJ.

***ALSQ**
Specifies MQ alias queue objects with names specified by OBJ.

***LCLQ**
Specifies MQ local queue objects with names specified by OBJ.

***MDLQ**
Specifies MQ model queues objects with names specified by OBJ.

***RMTQ**
Specifies MQ remote queue objects with names specified by OBJ.

***AUTHINFO**
Specifies MQ authentication information objects with names specified by OBJ.

***CTLG**
Specifies the MQ queue manager catalog object. This has the same name as the queue manager object.

***MQM**
Specifies the Message Queue Manager object.

***CHL** Specifies MQ channel objects with names specified by OBJ.

***CLTCN**
Specifies MQ client connection channel objects with names specified by OBJ.

***NMLIST**
Specifies MQ namelist objects with names specified by OBJ.

***PRC** Specifies MQ process objects with names specified by OBJ.

***LSR** Specifies MQ listener objects with names specified by OBJ.

***SVC** Specifies MQ service objects with names specified by OBJ.

***SYNCFILE**
Specifies the MQ channel synchronisation file.

***TOPIC**
Specifies the MQ topic objects with names specified by OBJ.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Display Journal Receiver Data (DSPJRNDTA)

Specifies whether additional messages should be written to the job log when the command completes to inform the user which journal receivers are still required by WebSphere MQ.

The possible values are:

***NO** No messages are written to the job log.

***YES** Messages will be sent to the job log when the command completes. The messages will contain details about which journal receivers are required by WebSphere MQ.

Examples

None

Error messages

Unknown

Recreate MQ Object (RCRMQMOBJ)

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

The Recreate MQ Object (RCRMQMOBJ) command is used to provide a recovery mechanism for damaged MQ objects. The command completely recreates the objects from information recorded in the MQ journals. If no damaged objects exist, no action is performed.

Parameters

Keyword	Description	Choices	Notes
OBJ	Object name	*Character value*, *ALL	Required, Positional 1

Keyword	Description	Choices	Notes
OBJTYPE	Object type	*ALL, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *CTLG, *MQM, *NMLIST, *PRC, *CHL, *CLTCN, *LSR, *SVC, *SYNCFILE, *CLCHLTAB, *TOPIC	Required, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3

Object name (OBJ)

Specifies the name of the objects which should be recreated if they are damaged. This is a 48-character MQ object or generic object name.

The possible values are:

***ALL** All damaged MQ objects of the specified type (OBJTYPE) are recreated.

generic-object-name
Specify the generic name of the objects to be recreated. A generic name is a character string followed by an asterisk (*). For example, ABC*. It selects all objects that have names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

object-name
The name of an MQ object to be recreated if it is damaged.

Object type (OBJTYPE)

Specifies the object type of the objects to be recreated.

The possible values are:

***ALL** Specifies all MQ object types.

***Q** Specifies MQ queue objects with names specified by OBJ.

***ALSQ**
Specifies MQ alias queue objects with names specified by OBJ.

***LCLQ**
Specifies MQ local queue objects with names specified by OBJ.

***MDLQ**
Specifies MQ model queues with names specified by OBJ.

***RMTQ**
Specifies MQ remote queue objects with names specified by OBJ.

***AUTHINFO**
Specifies MQ authentication information objects with names specified by OBJ.

***CTLG**
Specifies the message queue manager catalog object. The catalog object has the same name as the message queue manager object. It holds the names of MQ objects. A user needs authorities on this object to be able to start or stop the message queue manager, or, to create or delete MQ queues and process definitions.

***MQM**
Specifies the message queue manager. This object holds the attributes of the message queue manager.

***CHL** Specifies MQ channel objects with names specified by OBJ.

***CLTCN**
Specifies MQ client connection channel objects with names specified by OBJ.

***NMLIST**
Specifies MQ namelist objects with names specified by OBJ.

***PRC** Specifies MQ process objects with names specified by OBJ.

***LSR** Specifies MQ listener objects with names specified by OBJ.

***SVC** Specifies MQ service objects with names specified by OBJ.

***SYNCFILE**
Specifies the MQ channel synchronisation file.

***SYNCFILE**
Specifies the MQ client channel table file.

***TOPIC**
Specifies MQ topic objects with names specified by OBJ.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Examples

None

Error messages

Unknown

Refresh WebSphere MQ Authority (RFRMQMAUT)

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

The WebSphere MQ security cache refresh (RFRMQMAUT) command refreshes the WebSphere MQ Object Authority Manager security cache.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
TYPE	Refresh Type	***AUTHSERV** , *SSL	Optional, Positional 2

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager to perform the security refresh.

The possible values are:

queue-manager-name
Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

***DFT** Specifies that the default queue manager should be used.

Refresh Type (TYPE)

The type of security refresh to be performed. The possible values are:

***AUTHSERV**
Refreshes the list of authorizations held internally by the authorization services component.

***SSL** Refreshes the cached view of the SSL Key Repository allowing updates to become effective when the command has completed successfully. Also

refreshes the locations of the LDAP servers to be used for Certificate Revocation Lists and the Key Repository.

Examples

None

Error messages

Unknown

Refresh MQ Cluster (RFRMQMCL)

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

The Refresh MQ Cluster (RFRMQMCL) command refreshes locally held cluster information (including any autodefined channels that are in doubt), and forces it to be rebuilt. This enables you to perform a ″cold-start″ on the cluster.

Parameters

Keyword	Description	Choices	Notes
CLUSTER	Cluster Name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
REPOS	Refresh Repository	***NO**, *YES	Optional, Positional 3

Cluster Name (CLUSTER)

The name of the cluster to be refreshed.

The possible values are:

'*' The queue manager is refreshed in all of the clusters to which it belongs.

If Refresh Repository is also set to *YES, then the queue manager restarts its search for repository queue managers, using information in the local cluster-sender channel definitions.

name Specify the name of the cluster.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Refresh Repository (REPOS)

Specifies whether the information about repository queue managers should be refreshed.

The possible values are:

***NO** Do not refresh repository information.

***YES** Refresh repository information. This value cannot be specified if the queue manager is itself a repository manager.

Examples

None

Error messages

Unknown

Resume Cluster Queue Manager (RSMMQMCLQM)

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

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

Parameters

Keyword	Description	Choices	Notes
CLUSTER	Cluster Name	*Character value*	Optional, Positional 1
CLUSNL	Cluster Name List	*Character value*	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3

Cluster Name (CLUSTER)

Specifies the name of the cluster for which the queue manager is available for processing.

cluster-name
Specify the name of the cluster.

Cluster Name List (CLUSNL)

Specifies the namelist specifying a list of clusters for which the queue manager is available for processing.

namelist
Specify the name of the namelist.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Examples

None

Error messages

Unknown

Reset MQ Channel (RSTMQMCHL)

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

The Reset MQ Channel (RSTMQMCHL) command resets the message sequence number for an MQ channel to a specified sequence number for use the next time that the channel is started.

You are recommended to uset this command for Sender(*SDR), Server (*SVR) and Cluster-sender (*CLUSSDR) channels only.

If you use this command for a Receiver (*RCVR), Requester (*RQSTR) or Cluster-receiver (*CLUSRCVR) channel, the value at the other end of the channel is NOT reset. You must reset the values separately.

The command does not work for Server-connection (*SVRCN) channels.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*	Required, Positional 1
MSGSEQNUM	Message sequence number	1-999999999, **1**	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3

Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

channel-name
Specify the channel name.

Message sequence number (MSGSEQNUM)

Specifies the new message sequence number.

The possible values are:

1 The new message sequence number is 1.

message-sequence-number
Specify the new message sequence number ranging from 1 through 999999999.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

Reset Cluster (RSTMQMCL)

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

Use the Reset Cluster (RSTMQMCL) command to forcibly remove a queue manager from a cluster.

Parameters

Keyword	Description	Choices	Notes
CLUSTER	Cluster Name	*Character value*	Required, Positional 1
QMNAME	Queue Manager Name for removal	*Character value*, *QMID	Required, Positional 2
ACTION	Action	***FRCRMV**	Optional, Positional 3

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 4
QUEUES	Remove Queues	***NO**, *YES	Optional, Positional 5
QMID	Queue Manager Id for removal	*Character value*	Optional, Positional 6

Cluster Name (CLUSTER)

Specifies the name of cluster from which the queue manager is to be forcibly removed.

cluster-name
Specify the name of the cluster.

Queue Manager Name for removal (QMNAME)

Specifies the name of the queue manager to be forcibly removed.

The possible values are:

***QMID**
This enables you to specify the identifier of the queue manager to be forcibly removed.

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

Action (ACTION)

Specifies the action to take on the specified queue manager.

***FRCRMV**
Requests that the queue manager is forcibly removed from the cluster. This might be needed to ensure proper cleanup after a queue manager has been deleted. This action can be requested by a repository queue manager only.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Remove Queues (QUEUES)

Specifies whether cluster queues should be removed from the cluster.

The possible values are:

***NO** Do not remove the queues belonging to the queue manger being removed from the cluster.

***YES** Remove queues belonging to the queue manager being removed from the cluster.

Queue Manager Id for removal (QMID)

Specifies the identifier of the queue manager to be forcibly removed.

queue-manager-identifier
Specify the identifier of the queue manager.

Examples

None

Error messages

Unknown

Resolve MQ Channel (RSVMQMCHL)

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

The Resolve MQ Channel (RSVMQMCHL) command requests a channel to commit or backout in-doubt messages.

This command is used when the other end of a link fails during the confirmation period, and for some reason it is not possible to reestablish the connection.

In this situation, the sending end remains in an in-doubt state, about whether the messages were received. Any outstanding units of work need to be resolved with either backout or commit.

*BCK restores messages to the transmission queue and *CMT discards them.

Use this command for sender (*SDR) and server (*SVR) channels only.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*	Required, Positional 1
OPTION	Resolve option	*CMT, *BCK	Required, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3

Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

channel-name
Specify the channel name.

Resolve option (OPTION)

Specifies whether to back out or commit the messages.

The possible values are:

***CMT** The messages are committed, that is, they are deleted from the transmission queue.

***BCK** The messages are backed out, that is, they are restored to the transmission queue.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

RUNMQSC (RUNMQSC)

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

The Run WebSphere MQ Commands (RUNMQSC) command allows you to issue MQSC commands interactively for the specified queue manager.

Parameters

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

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

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

Examples

None

Error messages

Unknown

Revoke MQ Object Authority (RVKMQMAUT)

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

The Revoke MQ Authority (RVKMQMAUT) command is used to reset, or take away specific or all authority for the named objects from the users named in the command.

The RVKMQMAUT command can be used by anyone in the QMQMADM group, that is, anyone whose user profile specifies QMQMADM as a primary or supplemental group profile.

Parameters

Keyword	Description	Choices	Notes
OBJ	Object name	*Character value*	Required, Positional 1
OBJTYPE	Object type	*ALL, *Q, *ALSQ, *LCLQ, *MDLQ, *RMTQ, *AUTHINFO, *MQM, *NMLIST, *PRC, *LSR, *SVC, *CHL, *CLTCN, *TOPIC	Required, Positional 2
USER	User names	Single values: *PUBLIC Other values (up to 50 repetitions): *Name*	Required, Positional 3
AUT	Authority	Values (up to 21 repetitions): *ALTUSR, *BROWSE, *CONNECT, *GET, *INQ, *PUT, *SET, *PUB, *SUB, *RESUME, *PASSALL, *PASSID, *SETALL, *SETID, *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP, *ALL, *ALLADM, *ALLMQI, *REMOVE, *CTRL, *CTRLX	Required, Positional 4

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 5
SRVCOMP	Service Component name	*Character value*, ***DFT**	Optional, Positional 6

Object name (OBJ)

Specifies the name of the objects for which specific authorities are revoked.

The possible values are:

***ALL** All objects of the type specified by the value of the OBJTYPE parameter at the time the command is issued. *ALL cannot represent a generic profile.

object-name
Specify the name of an MQ object for which specific authority is given to one or more users.

generic profile
Specify the generic profile of the objects to be selected. A generic profile is a character string containing one or more generic characters anywhere in the string. This profile is used to match the object name of the object under consideration at the time of use. The generic characters are (?), (*) and (**).

? matches a single character in an object name.

* matches any string contained within a qualifier, where a qualifier is the string between fullstops (.). For example ABC* matches ABCDEF but not ABCDEF.XYZ.

** matches one or more qualifiers. For example ABC.**.XYZ matches ABC.DEF.XYZ and ABC.DEF.GHI.XYZ, ** can only appear once in a generic profile.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

Object type (OBJTYPE)

Specifies the type of the objects for which specific authorities are revoked.

***ALL** All MQ object types.

***Q** All queue object types.

***ALSQ**
Alias queue.

***LCLQ**
Local queue.

***MDLQ**
Model queue.

***RMTQ**
Remote queue.

***AUTHINFO**
Authentication Information object.

***MQM**
Message Queue Manager.

***NMLIST**
Namelist object.

***PRC** Process definition.

***CHL** Channel object.

***CLTCN**
Client Connection Channel object.

***LSR** Listener object.

***SVC** Service object.

***TOPIC**
Topic object.

User names (USER)

Specifies the user names of one or more users whose specific authorities to the named object are being removed. If a user was given the authority by USER(*PUBLIC) being specified in the Grant MQ Authority (GRTMQMAUT) command, the same authorities are revoked by *PUBLIC being specified in this parameter. Users given specific authority by having their names identified in the GRTMQMAUT command must have their names specified on this parameter to remove the same authorities.

The possible values are:

***PUBLIC**
The specified authorities are taken away from users who do not have specific authority for the object, who are not on the authorization list, and whose user group has no authority. Users who have specific authority still retain their authorities to the object.

user-profile-name
Specify the user names of one or more users who are having the specified authorities revoked. The authorities listed in the AUT parameter are being specifically taken away from each identified user. This parameter cannot be used to remove public authority from specific users; only authorities that were specifically given to them can be specifically revoked. You can specify up to 50 user profile names.

Authority (AUT)

Specifies the authority being reset or taken away from the users specified in the USER parameter. You can specify values for AUT as a list of specific and general authorities in any order, where the general authorities can be:

*REMOVE, which deletes the profile. It is not the same as *ALL, because *ALL leaves the profile in existence with no authorities. *REMOVE cannot be specified with user QMQMADM unless the object is a generic profile.

*ALL, which confers all authorities to the specified users.

*ALLADM, which confers all of *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP, *CTRL and *CTRLX.

*ALLMQI, which confers all of *ALTUSR, *BROWSE, *CONNECT, *GET, *INQ, *PUT, *SET, *PUB, *SUB and *RESUME.

Authorizations for different object types

***ALL** All authorizations. Applies to all objects.

***ADMCHG**
Change an object. Applies to all objects.

***ADMCLR**
Clear a queue. Applies to queues only.

***ADMCRT**
Create an object. Applies to all objects.

***ADMDLT**
Delete an object. Applies to all objects.

***ADMDSP**
Display the attributes of an object. Applies to all objects.

***ALLADM**
Perform administration operations on an object. Applies to all objects.

***ALLMQI**
Use all MQI calls applicable to an object. Applies to all objects.

***ALTUSR**
Allow another user's authority to be used for MQOPEN and MQPUT1 calls. Applies to queue manager objects only.

***BROWSE**
Retrieve a message from a queue by issuing an MQGET call with the BROWSE option. Applies to queue objects only.

***CONNECT**
Connect the application to a queue manager by issuing an MQCONN call. Applies to queue manager objects only.

***CTRL**
Control startup and shutdown of channels, listeners and services.

***CTRLX**
Reset sequence number and resolve indoubt channels.

***GET** Retrieve a message from a queue using an MGET call. Applies to queue objects only.

***INQ** Make an inquiry on an object using an MQINQ call. Applies to all objects.

***PASSALL**
Pass all context on a queue. Applies to queue objects only.

***PASSID**
Pass identity context on a queue. Applies to queue objects only.

***PUT** Put a message on a queue using an MQPUT call. Applies to queue objects only.

***SET** Set the attributes of an object using an MQSET call. Applies to queue, queue manager, and process objects only.

***SETALL**
Set all context on an object. Applies to queue and queue manager objects only.

***SETID**
Set identity context on an object. Applies to queue and queue manager objects only.

Authorizations for MQI calls

***ALTUSR**
Allow another user's authority to be used for MQOPEN and MQPUT1 calls.

***BROWSE**
Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.

***CONNECT**
Connect the application to the specified queue manager by issuing an MQCONN call.

***GET** Retrieve a message from a queue by issuing an MQGET call.

***INQ** Make an inquiry on a specific queue by issuing an MQINQ call.

***PUT** Put a message on a specific queue by issuing an MQPUT call.

***SET** Set attributes on a queue from the MQI by issuing an MQSET call.

***PUB** Open a topic to publish a message using the MQPUT call.

***SUB** Create, Alter or Resume a subscription to a topic using the MQSUB call.

***RESUME**
Resume a subscription using the MQSUB call.

If you open a queue for multiple options, you must be authorized for each of them.

Authorizations for context

***PASSALL**
Pass all context on the specified queue. All the context fields are copied from the original request.

***PASSID**
Pass identity context on the specified queue. The identity context is the same as that of the request.

***SETALL**
Set all context on the specified queue. This is used by special system utilities.

***SETID**
Set identity context on the specified queue. This is used by special system utilities.

Authorizations for MQSC and PCF commands

***ADMCHG**
Change the attributes of the specified object.

***ADMCLR**
Clear the specified queue (PCF Clear queue command only).

***ADMCRT**
Create objects of the specified type.

***ADMDLT**
Delete the specified object.

***ADMDSP**
Display the attributes of the specified object.

***CTRL**
Control startup and shutdown of channels, listeners and services.

***CTRLX**
Reset sequence number and resolve indoubt channels.

Authorizations for generic operations

***ALL** Use all operations applicable to the object.

***ALLADM**
Perform all administration operations applicable to the object.

***ALLMQI**
Use all MQI calls applicable to the object.

***REMOVE**
Delete the authority profile to the specified object.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Service Component name (SRVCOMP)

Specifies the name of the installed authorization service to which the authorizations apply.

The possible values are:

***DFT** Use the first installed authorization component.

Authorization-service-component-name
The component name of the required authorization service as specified in the Queue Manager's qm.ini file.

Examples

None

Error messages

Unknown

Suspend Cluster Queue Manager (SPDMQMCLQM)

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

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

Parameters

Keyword	Description	Choices	Notes
CLUSTER	Cluster Name	*Character value*	Optional, Positional 1
CLUSNL	Cluster Name List	*Character value*	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
MODE	Mode	***QUIESCE**, *FORCE	Optional, Positional 4

Cluster Name (CLUSTER)

Specifies the name of the cluster for which the queue manager is no longer available for processing.

cluster-name
Specify the name of the cluster.

Cluster Name List (CLUSNL)

Specifies the name of the namelist specifying a list of clusters for which the queue manager is no longer available for processing.

namelist
Specify the name of the namelist.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Mode (MODE)

Specifies how the suspension of availability is to take effect:

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

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

Examples

None

Error messages

Unknown

Start Message Queue Manager (STRMQM)

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

The Start Message Queue Manager (STRMQM) command starts the local queue manager.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
RDEFSYS	Redefine system objects	*YES, ***NO**	Optional, Positional 2
STRSTSDTL	Startup Status Detail	***ALL**, *MIN	Optional, Positional 3
STRSVC	Service startup	***YES**, *NO	Optional, Positional 4
REPLAY	Perform replay only	*YES, ***NO**	Optional, Positional 5
ACTIVATE	Activate backup	*YES, ***NO**	Optional, Positional 6

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Redefine system objects (RDEFSYS)

Specifies whether the default and system objects are redefined.

***NO** Do not redefine the system objects.

***YES** Starts the queue manager, redefines the default and system objects, then stops the queue manager. Any existing system and default objects belonging to the queue manager are replaced if you specify this flag.

Startup Status Detail (STRSTSDTL)

Specifies the detail of status messages that are issued whilst starting the queue manager.

***ALL** Display all startup status messages. This level of detail includes periodically displaying messages detailing transaction recovery and log replay. This level of detail can be useful in tracking queue manager startup progress following the abnormal termination of a queue manager.

***MIN** Displays a minimum level of status messages.

Service startup (STRSVC)

Specifies whether the additional following QMGR components are started when the Queue Manager is started:

- The Channel Initiator
- The Command Server
- Listeners with CONTROL set to QMGR or STARTONLY
- Services with CONTROL set to QMGR or STARTONLY

***YES** Start the channel initiator, command server, listeners and services when the Queue Manager is started.

***NO** Do not start the channel initiator, command server, listeners or services when the Queue Manager is started.

Perform replay only (REPLAY)

Whether the Queue Manager is being started to perform replay only. This enables a backup copy of a Queue Manager on a remote machine to replay logs created by the corresponding active machine, and to allow the backup Queue Manager to be activated in the event of a disaster on the active machine.

***NO** The Queue Manager is not being started to perform replay only.

***YES** The Queue Manager is being started to perform replay only. The STRMQM command will end when replay is complete.

Activate backup (ACTIVATE)

Specifies whether to mark a Queue Manager as active. A Queue Manager that has been started with the REPLAY option is marked as a backup Queue Manager and cannot be started before it has been activated.

***NO** The Queue Manager is not to be marked as active.

***YES** The Queue Manager is to be marked as active. Once a Queue Manager has been activated then it can be started as a normal Queue Manager using the STRMQM command without the REPLAY and ACTIVATE options.

Examples

None

Error messages

Unknown

Start MQ Pub/Sub Broker (STRMQMBRK)

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

The Start WebSphere MQ broker (STRMQMBRK) command starts a broker for a specified queue manager.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*	Required, Positional 1
PARENTMQM	Parent Message Queue Manager	*Character value*	Optional, Positional 2

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

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

Parent Message Queue Manager (PARENTMQM)

Specifies the name of the queue manager that provides the parent broker function. Before you can add a broker to the network, channels in both directions must exist between the queue manager that hosts the new broker, and the queue manager that hosts the parent.

On restart, this parameter is optional. If present, it must be the same as it was when previously specified. If this is the root-node broker, the queue manager specified becomes its parent. You cannot specify the name of the parent broker when you use triggering to start a broker.

After a parent has been specified, it is only possible to change parentage in exceptional circumstances in conjunction with the CLRMQMBRK command. By changing a root node to become the child of an existing broker, two hierarchies can be joined. This causes subscriptions to be propagated across the two hierarchies,

which now become one. After that, publications start to flow across them. To ensure predictable results, it is essential that you quiesce all publishing applications at this time.

If the changed broker detects a hierarchical error (that is, if the new parent is found also to be a descendant), it immediately shuts down. The administrator must then use CLRMQMBRK at both the changed broker and the new, false parent to restore the previous status. A hierarchical error is detected by propagating a message up the hierarchy, which can complete only when the relevant brokers and links are available.

Examples

None

Error messages

Unknown

Start MQ Channel (STRMQMCHL)

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

The Start MQ Channel (STRMQMCHL) command starts an MQ channel.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

channel-name
Specify the channel name.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

Start MQ Channel Initiator (STRMQMCHLI)

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

The Start MQ Channel Initiator (STRMQMCHLI) command starts an MQ channel initiator.

Parameters

Keyword	Description	Choices	Notes
QNAME	Queue name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Queue name (QNAME)

Specifies the name of the initiation queue for the channel initiation process. That is, the initiation queue that is specified in the definition of the transmission queue.

The possible values are:

queue-name
Specify the name of the initiation queue.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

Start MQ Command Server (STRMQMCSVR)

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

The Start MQ Command Server (STRMQMCSVR) command starts the MQ command server for the specified queue manager.

Parameters

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

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

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

Examples

None

Error messages

Unknown

Start WebSphere MQ DLQ Handler (STRMQMDLQ)

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

Use the Start WebSphere MQ Dead-Letter Queue Handler (STRMQMDLQ) command to perform various actions on selected messages. The command specifies a set of rules that can both select a message and perform the action on that message.

The STRMQMDLQ command takes its input from the rules table as specified by SRCFILE and SRCMBR. When the command processes, the results and a summary are written to the printer spooler file.

Note:

The WAIT keyword, defined in the rules table, determines whether the dead-letter queue handler ends immediately after processing messages, or waits for new messages to arrive.

Parameters

Keyword	Description	Choices	Notes
UDLMSGQ	Undelivered message queue	*Character value*, *DFT, *NONE	Required, Positional 1
SRCMBR	Member containing input	*Name*, *FIRST	Required, Positional 2
SRCFILE	Input file	*Qualified object name*	Optional, Positional 3
	Qualifier 1: Input file	*Name*, **QTXTSRC**	
	Qualifier 2: Library	*Name*, ***LIBL** , *CURLIB	
MQMNAME	Message Queue Manager name	*Character value*, ***DFT** , *NONE	Optional, Positional 4

Undelivered message queue (UDLMSGQ)

Specifies the name of the local undelivered message queue that is to be processed.

The possible values are:

***DFT** The local undelivered-message queue used is taken from the default queue manager for the installation. If this option is specified, the INPUTQ keyword stated in the rules table will be overriden by the default undelivered-message queue for the queue manager.

undelivered-message-queue-name
Specify the name of the local undelivered-message queue to be used. If this option is specified, the INPUTQ keyword stated in the rules table will be overriden by the stated undelivered-message queue.

***NONE**
The queue that is named by the INPUTQ keyword in the rules table is used, or the system-default dead-letter queue if the INPUTQ keyword in the rules table is blank.

Member containing input (SRCMBR)

Specifies the name of the source member, containing the user-written rules table to be processed.

The possible values are:

***FIRST**
The first member of the file is used.

source-member-name
Specify the name of the source member.

Input file (SRCFILE)

Specifies the name of the source file and library, in the form LIBRARY/FILE, that contains the user-written rules table to be processed.

The possible values are:

***LIBL** Search the library list for the file name.

***CURLIB**
Use the current library.

source-library-name
Specify the name of the library that is being used.

The possible values are:

QTXTSRC
Use QTXTSRC.

source-file-name
Specify the name of the source file.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

***NONE**
The queue manager that is named by the INPUTQM keyword in the rules table is used, or the system-default queue manger if the INPUTQM keyword in the rules table is blank.

Examples

None

Error messages

Unknown

Start MQ Listener (STRMQMLSR)

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

The Start MQ Listener (STRMQMLSR) command starts an MQ TCP/IP listener.

This command is valid for TCP/IP transmission protocols only.

You can specify either a listener object or specific listener attributes.

Parameters

Keyword	Description	Choices	Notes
PORT	Port number	1-65535, ***DFT**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
IPADDR	IP Address	*Character value*, ***DFT**	Optional, Positional 3
BACKLOG	Listener backlog	0-999999999, ***DFT**	Optional, Positional 4
LSRNAME	Listener name	*Character value*, ***NONE**	Optional, Positional 5

Port number (PORT)

The port number to be used by the listener.

The possible values are:

***DFT** Port number 1414 is used.

port-number
The port number to be used.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

IP Address (IPADDR)

The IP address to be used by the listener.

The possible values are:

***DFT** The listener will listen on all IP addresses available to the TCP/IP stack.

ip-addr
The IP address to be used.

Listener backlog (BACKLOG)

The number of concurrent connection requests the listener supports.

The possible values are:

***DFT** 255 concurrent connection requests are supported.

backlog
The number of concurrent connection requests supported.

Listener name (LSRNAME)

The name of the MQ listener object to be started.

The possible values are:

***NONE**
No listener object is specified.

listener-name
Specify the name of the listener object to be started.

Examples

None

Error messages

Unknown

Start WebSphere MQ Commands (STRMQMMQSC)

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

The Start WebSphere MQ Commands (STRMQMMQSC) command initiates a set of WebSphere MQ Commands (MQSC) and writes a report to the printer spooler file.

Each report consists of the following elements:

- A header identifying MQSC as the source of the report.
- A numbered listing of the input MQSC commands.
- A syntax error message for any commands in error.
- A message indicating the outcome of running each correct command.
- Other messages for general errors running MQSC, as needed.
- A summary report at the end.

Parameters

Keyword	Description	Choices	Notes
SRCMBR	Member containing input	*Name*, *FIRST	Required, Positional 1
SRCFILE	Input file	*Qualified object name*	Optional, Positional 2
	Qualifier 1: Input file	*Name*, **QMQSC**	
	Qualifier 2: Library	*Name*, ***LIBL** , *CURLIB	
OPTION	Option	***RUN** , *VERIFY, *MVS	Optional, Positional 3
WAIT	Wait time	1-999999	Optional, Positional 4
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 5

Member containing input (SRCMBR)

Specifies the name of the source member, containing the MQSC, to be processed.

The possible values are:

source-member-name
Specify the name of the source member.

***FIRST**
The first member of the file is used.

Input file (SRCFILE)

Specifies the qualified name of the file, in the form LIBRARY/FILE, that contains the MQSC to be processed.

The possible values are:

***LIBL** The library list is searched for the file name.

***CURLIB**
The current library is used.

source-library-name
Specify the name of the library to be used.

The possible values are:

QMQSC
QMQSC is used.

source-file-name
Specify the name of the source file.

Option (OPTION)

Specifies how the MQSC commands are to be processed.

The possible values are:

***RUN**
If this value is specified and a value for the WAIT parameter is not specified the MQSC commands are processed directly by the local queue manager. If this value is specified and a value is also specified for the WAIT parameter the MQSC commands are processed indirectly by a remote queue manager,

***VERIFY**
The MQSC commands are verified and a report is written, but the commands are not run.

***MVS** The MQSC commands are processed indirectly by a remote queue manager running under MVS/ESA™. If you specify this option you must also specify a value for the WAIT parameter.

Wait time (WAIT)

Specifies the time in seconds that the STRMQMMQSC command waits for replies to indirect MQSC commands. Specifying a value for this parameter indicates that MQSC commands are executed in indirect mode by a remote queue manager. Specifying a value for this parameter is only valid when the OPTION parameter is specified as *RUN or *MVS.

In indirect mode, MQSC commands are queued on the command queue of a remote queue manager. Reports from the commands are then returned to the local queue manager specified in MQMNAME. Any replies received after this time are discarded, however, the MQSC command is still run.

The possible values are:

1 - 999999
Specify the waiting time in seconds.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Examples

None

Error messages

Unknown

Start MQ Service (STRMQMSVC)

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

The Start MQ Service (STRMQMSVC) command starts an MQ service.

Parameters

Keyword	Description	Choices	Notes
SVCNAME	Service name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Service name (SVCNAME)

The name of the MQ service object to be started.

The possible values are:

***NONE**
No service object is specified.

service-name
Specify the name of the service definition. The maximum length of the string is 48 bytes.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

Start MQ Trigger Monitor (STRMQMTRM)

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

The Start MQ Trigger Monitor (STRMQMTRM) command starts the MQ trigger monitor for the specified queue manager.

Parameters

Keyword	Description	Choices	Notes
INITQNAME	Initiation queue	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Initiation queue INITQNAME

Specifies the name of the initiation queue.

initiation-queue-name
Specify the name of the initiation queue

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Examples

None

Error messages

Unknown

Trace MQ (TRCMQM)

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

The Trace MQ (TRCMQM) command controls tracing for all MQ jobs. TRCMQM, which sets tracing on or off, can trace message queue interface (MQI) functions, function flow, and WebSphere MQ for i5/OS components together with any messages issued by WebSphere MQ.

Parameters

Keyword	Description	Choices	Notes
TRCEARLY	Trace early	***NO** , *YES	Optional, Positional 1
SET	Trace option setting	***ON** , *OFF, *END	Optional, Positional 2
OUTPUT	Output	***MQM** , *MQMFMT, *PEX, *ALL	Optional, Positional 3
TRCLEVEL	Trace level	***DFT** , *DETAIL, *PARMS	Optional, Positional 4
TRCTYPE	Trace types	Single values: ***ALL** Other values (up to 14 repetitions): *API, *CMTRY, *COMMS, *CSDATA, *CSFLOW, *LQMDATA, *LQMFLOW, *OTHDATA, *OTHFLOW, *RMTDATA, *RMTFLOW, *SVCDATA, *SVCFLOW, *VSNDATA	Optional, Positional 5
EXCLUDE	Exclude types	Single values: ***NONE** Other values (up to 14 repetitions): *API, *CMTRY, *COMMS, *CSDATA, *CSFLOW, *LQMDATA, *LQMFLOW, *OTHDATA, *OTHFLOW, *RMTDATA, *RMTFLOW, *SVCDATA, *SVCFLOW, *VSNDATA	Optional, Positional 6
MAXSTG	Maximum storage to use	1-16, ***DFT**	Optional, Positional 7
DATASIZE	Trace data size	1-99999999, ***DFT** , *ALL, *NONE	Optional, Positional 8
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 9
TRCDIR	Trace directory	*Character value*, ***DFT**	Optional, Positional 10

Keyword	Description	Choices	Notes
JOB	Job information	Values (up to 8 repetitions): *Element list*	Optional, Positional 11
	Element 1: Job name	*Qualified job name*	
	Qualifier 1: Job name	*Generic name, name*	
	Qualifier 2: User	*Character value*	
	Qualifier 3: Number	*Character value*	
	Element 2: Thread identifier	*Character value*	
STRCTL	Trace start control	Values (up to 8 repetitions): *Character value*	Optional, Positional 12
ENDCTL	Trace end control	Values (up to 8 repetitions): *Character value*	Optional, Positional 13

Trace early (TRCEARLY)

Specifies whether early tracing is selected.

Early tracing applies to all jobs for all queue managers. If a queue manager is not currently active or does not exist, then early trace will become effective during start-up or creation.

***NO** Early tracing is not enabled.

***YES** Early tracing is enabled.

Trace option setting (SET)

Specifies the collection of trace records.

The possible values are:

***ON** The collection of trace records is started.

For TRCEARLY(*NO), the collection of trace records will not be started until after the queue manager is available.

***OFF** The collection of trace records is stopped. Trace records are written to files in /QIBM/UserData/mqm/trace

***END** The collection of trace records is stopped for all queue managers.

Output (OUTPUT)

Identifies the type of trace output that this command applies.

The possible values are:

***MQM**
This command applies to the collection of binary WebSphere MQ trace output in the directory specified by the TRCDIR parameter.

***MQMFMT**
This command applies to the collection of formatted WebSphere MQ trace output in the directory specified by the TRCDIR parameter.

***PEX** This command applies to the collection of Performance Explorer (PEX) trace output.

***ALL** This option applies to the collection of both WebSphere MQ unformatted trace and PEX trace output.

Trace level (TRCLEVEL)

Activates tracing level for flow processing trace points.

The possible values are:

***DFT** Activates tracing at default level for flow processing trace points.

***DETAIL**
Activates tracing at high-detail level for flow processing trace points.

***PARMS**
Activates tracing at default-detail level for flow processing trace points.

Trace types (TRCTYPE)

Specifies the type of trace data to store in the trace file. If this parameter is omitted, all trace points are enabled.

The possible values are:

***ALL** All the trace data as specified by the following keywords is stored in the trace file.

trace-type-list
You can specify more than one option from the following keywords, but each option can appear only once.

***API** Output data for trace points associated with the MQI and major queue manager components.

***CMTRY**
Output data for trace points associated with comments in the MQ components.

***COMMS**
Output data for trace points associated with data flowing over communications networks.

***CSDATA**
Output data for trace points associated with internal data buffers in common services.

***CSFLOW**
Output data for trace points associated with processing flow in common services.

***LQMDATA**
Output data for trace points associated with internal data buffers in the local queue manager.

***LQMFLOW**
Output data for trace points associated with processing flow in the local queue manager.

***OTHDATA**
Output data for trace points associated with internal data buffers in other components.

***OTHFLOW**
Output data for trace points associated with processing flow in other components.

***RMTDATA**
Output data for trace points associated with internal data buffers in the communications component.

***RMTFLOW**
Output data for trace points associated with processing flow in the communications component.

***SVCDATA**
Output data for trace points associated with internal data buffers in the service component.

***SVCFLOW**
Output data for trace points associated with processing flow in the service component.

***VSNDATA**
Output data for trace points associated with the version of WebSphere MQ running.

Exclude types (EXCLUDE)

Specifies the type of trace data to omit from the trace file. If this parameter is omitted, all trace points specified in TRCTYPE are enabled.

The possible values are:

***ALL** All the trace data as specified by the following keywords is stored in the trace file.

trace-type-list
You can specify more than one option from the following keywords, but each option can appear only once.

***API** Output data for trace points associated with the MQI and major queue manager components.

***CMTRY**
Output data for trace points associated with comments in the MQ components.

***COMMS**
Output data for trace points associated with data flowing over communications networks.

***CSDATA**
Output data for trace points associated with internal data buffers in common services.

***CSFLOW**
Output data for trace points associated with processing flow in common services.

***LQMDATA**
Output data for trace points associated with internal data buffers in the local queue manager.

***LQMFLOW**
Output data for trace points associated with processing flow in the local queue manager.

***OTHDATA**
Output data for trace points associated with internal data buffers in other components.

***OTHFLOW**
Output data for trace points associated with processing flow in other components.

***RMTDATA**
Output data for trace points associated with internal data buffers in the communications component.

***RMTFLOW**
Output data for trace points associated with processing flow in the communications component.

***SVCDATA**
Output data for trace points associated with internal data buffers in the service component.

***SVCFLOW**
Output data for trace points associated with processing flow in the service component.

***VSNDATA**
Output data for trace points associated with the version of WebSphere MQ running.

Maximum storage to use (MAXSTG)

Specifies the maximum size of storage to be used for the collected trace records.

The possible values are:

***DFT** The default maximum is 1 megabyte (1024 kilobytes).

maximum-megabytes
Specify a value ranging from 1 through 16.

Trace data size (DATASIZE)

Specifies the number of bytes of user data included in the trace.

The possible values are:

***DFT** The default trace value is used.

***ALL** All the user data is traced.

***NONE**
This option will turn off the trace for sensitive customer data.

data-size-in-bytes
Specify a value in ranging from 1 through 99999999.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

This parameter is only valid when TRCEARLY is set to *NO.

When TRCEARLY is set to *YES all queue managers are traced.

The possible values are:

***DFT** Trace the default queue manager.

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

Trace directory (TRCDIR)

Specifies the IFS directory for saving trace files for output type *MQM or *MQMFMT.

The possible values are:

***DFT** WebSphere MQ trace will be written into the '/QIBM/UserData/mqm/trace' directory.

trace-directory
Specify the name of the directory into which trace is written. If the name commences with '/' then the path name is based upon the root directory, otherwise the directory name is based upon the default WebSphere MQ directory.

Job information (JOB)

Specifies which jobs are to be traced.

Trace start control (STRCTL)

Specifies that trace is started when any FDCs with the specified FDC probeIDs are generated.

Trace end control (ENDCTL)

Specifies that trace is ended when any FDCs with the specified FDC probeIDs are generated.

Alternatively specifies that trace is ended after n seconds.

Examples

None

Error messages

Unknown

Work with MQ Queue Manager (WRKMQM)

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

The Work with Queue Managers (WRKMQM) command allows you to work with one or more queue manager definitions, and allows you to perform the following operations:

- Change a queue manager
- Create a queue manager
- Delete a queue manager
- Start a queue manager
- Display a queue manager
- End a queue manager
- Work with channels of a queue manager

- Work with namelists of a queue manager
- Work with queues of a queue manager
- Work with processes of a queue manager

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***ALL**	Optional, Positional 1

Message Queue Manager name (MQMNAME)

Specifies the name or names of the message queue managers to select.

The possible values are:

***ALL** All queue managers are selected.

generic-queue-manager-name
Specify the generic name of the queue managers to select. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all queue managers having names that start with the character string. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Note: You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered. You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

queue-manager-name
Specify the name of the queue manager. The name can contain up to 48 characters. The maximum number of characters is reduced if the system is using a double byte character set (DBCS).

Examples

None

Error messages

Unknown

Work with MQ Authority (WRKMQMAUT)

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

The Work with MQ Authority (WRKMQMAUT) displays a list of all the authority profile names and their types, which match the specified parameters. This enables you to delete, work with and create the authority records for an MQM authority profile record.

Parameters

Keyword	Description	Choices	Notes
OBJ	Object/Profile name	*Character value*, ***ALL**	Optional, Positional 1
OBJTYPE	Object type	*Q, *PRC, *MQM, *NMLIST, *AUTHINFO, *LSR, *SVC, *CHL, *CLTCN, ***ALL**, *TOPIC	Optional, Positional 2
OUTPUT	Output	*, *PRINT	Optional, Positional 3
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 4
SRVCOMP	Service Component name	*Character value*, ***DFT**	Optional, Positional 5

Object name (OBJ)

Specify the object name or authority profile name of the object to select.

The possible values are:

***ALL** All authority records matching the specified object type are listed. *ALL cannot represent a generic profile.

object-name
Specify the name of an MQ object; all authority records whose object name or generic profile name match this object name are selected.

generic profile
Specify the generic profile of an MQ object; only the authority record which exactly matches the generic profile is selected. A generic profile is a character string containing one or more generic characters anywhere in the string. The generic characters are (?), (*) and (**).

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

Object type (OBJTYPE)

Specifies the object type of the authority profile to select.

***ALL** All MQ object types.

***Q** All queue object types.

***AUTHINFO**
Authentication Information object.

***MQM**
Message Queue Manager.

***NMLIST**
Namelist object.

***PRC** Process definition.

***CHL** Channel object.

***CLTCN**
Client Connection Channel object.

***LSR** Listener object.

***SVC** Service object.

***TOPIC**
Topic object.

Output (OUTPUT)

Specifies whether the output from the command is shown at the requesting work station, or printed with the job's spooled output.

The possible values are:

***** Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**
A detailed list of the users and their authorities registered with the selected authority profile record is printed with the job's spooled output.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Service Component name (SRVCOMP)

Specify the name of the installed authorization service in which to search for the authorities to display.

The possible values are:

***DFT** All installed authorization components are searched for the specified authority profile name and object type.

Authorization-service-component-name
The component name of the authorization service as specified in the Queue Manager's qm.ini file.

Examples

None

Error messages

Unknown

Work with MQ Authority Data (WRKMQMAUTD)

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

The Work with MQ Authority Records (WRKMQMAUTD) displays a list of all the users registered to a particular authority profile name and type. This enables you to grant, revoke, delete and create authority records.

Parameters

Keyword	Description	Choices	Notes
OBJ	Object/Profile name	*Character value*	Required, Positional 1
OBJTYPE	Object type	*Q, *PRC, *MQM, *NMLIST, *AUTHINFO, *CHL, *CLTCN, *SVC, *LSR, *TOPIC	Required, Positional 2

Keyword	Description	Choices	Notes
USER	User name	*Name*, *PUBLIC, ***ALL**	Optional, Positional 3
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 4
SRVCOMP	Service Component name	*Character value*, ***DFT**	Optional, Positional 5

Object name (OBJ)

Specify the object name or authority profile name of the object to select.

object-name
Specify the name of an MQ object; all authority records whose object name or generic profile name match this object name are selected.

generic profile
Specify the generic profile of an MQ object; only the authority record which exactly matches the generic profile is selected. A generic profile is a character string containing one or more generic characters anywhere in the string. The generic characters are (?), (*) and (**).

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

Object type (OBJTYPE)

Specifies the object type of the authority profile to select.

***Q** All queue object types.

***AUTHINFO**
Authentication Information object.

***MQM**
Message Queue Manager.

***NMLIST**
Namelist object.

***PRC** Process definition.

***CHL** Channel object.

***CLTCN**
Client Connection Channel object.

***LSR** Listener object.

***SVC** Service object.

***TOPIC**
Topic object.

User name (USER)

Specifies the name of the user for whom authorities for the named object are displayed.

The possible values are:

***ALL** List all relevant users.

***PUBLIC**
The user name implying all users of the system.

user-profile-name
Specify the name of the user.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Service Component name (SRVCOMP)

Specify the name of the installed authorization service in which to search for the authorities to display.

The possible values are:

***DFT** All installed authorization components are searched for the specified authority profile name and object type.

Authorization-service-component-name
The component name of the authorization service as specified in the Queue Manager's qm.ini file.

Examples

None

Error messages

Unknown

Work with AuthInfo objects (WRKMQMAUTI)

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

The Work with MQ AuthInfo objects (WRKMQMAUTI) command allows you to work with multiple authentication information objects which are defined on the local queue manager.

This enables you to change, copy, create, delete, display, and display and change authority to an MQ authentication information object.

Parameters

Keyword	Description	Choices	Notes
AINAME	AuthInfo name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *ALTTIME, *AUTHTYPE, *CONNAME, *TEXT, *USERNAME	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

AuthInfo name (AINAME)

The name or names of the authentication information objects.

The possible values are:

***ALL or ***
All authentication information objects are selected.

generic-authinfo-name
The generic name of the authentication information objects. A generic name

is a character string followed by an asterisk (*). For example ABC*, it selects all authentication information objects having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

authentication-information-name
Specify the name of a single authentication information object.

Message Queue Manager name (MQMNAME)

The name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of an existing message queue manager. The maximum string length is 48 characters.

Filter command (WHERE)

This parameter can be used to selectively display those AuthInfo objects with particular AuthInfo attributes only.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***ALTDATE**
The date on which the definition or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

***ALTTIME**
The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

***AUTHTYPE**
The type of the authentication information object.

The filter value is one of the following:

***CRLLDAP**
The type of the authentication information object is CRLLDAP.

***CONNAME**
The address of the host on which the LDAP server is running.

The filter value is the address name.

***TEXT**
Descriptive comment.

The filter value is the text description of the queue.

***USERNAME**
The distinguished name of the user.

The filter value is the distinguised name.

Examples

None

Error messages

Unknown

Work with MQ Channels (WRKMQMCHL)

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

The Work with WebSphere MQ Channels (WRKMQMCHL) command allows you to work with one or more channel definitions. This enables you to create, start, end, change, copy, delete, ping, display and reset channels, and resolve in-doubt units of work.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*, ***ALL**	Optional, Positional 1
CHLTYPE	Channel type	*RCVR, *SDR, *SVR, *RQSTR, *SVRCN, *CLUSSDR, *CLUSRCVR, *CLTCN, ***ALL**	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
STATUS	Channel status	***ALL** , *INACTIVE, *STOPPED, *BINDING, *RETRYING, *RUNNING	Optional, Positional 4

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 5
	Element 1: Filter keyword	*AFFINITY, *ALTDATE, *ALTTIME, *BATCHHB, *BATCHINT, *BATCHSIZE, *CLNTWGHT, *CLUSNL, *CLUSTER, *CLWLPRTY, *CLWLRANK, *CLWLWGHT, *COMPHDR, *COMPMSG, *CONNAME, *CVTMSG, *DSCITV, *HRTBTINTVL, *KAINT, *LOCLADDR, *LONGRTY, *LONGTMR, *MAXINST, *MAXINSTC, *MAXMSGLEN, *MCANAME, *MCATYPE, *MCAUSRID, *MODENAME, *MONCHL, *MSGEXIT, *MSGRTYDATA, *MSGRTYEXIT, *MSGRTYITV, *MSGRTYNBR, *MSGUSRDATA, *NETPRTY, *NPMSPEED, *PROPCTL, *PUTAUT, *RCVEXIT, *RCVUSRDATA, *SCYEXIT, *SCYUSRDATA, *SEQNUMWRAP, *SHARECNV, *SHORTRTY, *SHORTTMR, *SNDEXIT, *SNDUSRDATA, *SSLCAUTH, *SSLCIPH, *SSLPEER, *STATCHL, *TEXT, *TGTMQMNAME, *TMQNAME, *TPNAME, *TRPTYPE, *USERID	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Channel name (CHLNAME)

Specifies the name or names of the WebSphere MQ channel definitions to be selected.

The possible values are:

***ALL** All channel definitions are selected.

generic-channel-name
Specify the generic name of the channel definitions to be selected. A generic name is a character string followed by an asterisk (*). For example ABC*, it selects all channel definitions having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

channel-name
Specify the name of the channel definition.

Channel type (CHLTYPE)

Specifies the type of channel definitions that are to be displayed.

The possible values are:

***ALL** All the channel types are selected.

***SDR** Sender channel

***SVR** Server channel

***RCVR**
Receiver channel

***RQSTR**
Requester channel

***SVRCN**
Server-connection channel

***CLUSSDR**
Cluster-sender channel

***CLUSRCVR**
Cluster-receiver channel

***CLTCN**
Client-connection channel

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Channel status (STATUS)

Specifies the status type of the WebSphere MQ channel definitions to be selected.

The possible values are:

***ALL** Channels with any status are selected.

***INACTIVE**
Only channels with an inactive status are selected.

***STOPPED**
Only channels with a stopped status are selected.

***BINDING**
Only channels with a binding status are selected.

***RETRYING**
Only channels with a retrying status are selected.

***RUNNING**
Only channels with a running status are selected.

Filter command (WHERE)

This parameter can be used to selectively display those channels with particular channel attributes only.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***AFFINITY**
Connection Affinity.

The filter value is one of the following:

***PREFERRED**
Preferred connection affinity.

***NONE**
No connection affinity.

***ALTDATE**
The date on which the definition or information was last altered.

The filter value is the data in the form yyyy-mm-dd.

***ALTTIME**
The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

***BATCHHB**
Batch heartbeat interval in milliseconds.

The filter value is the integer interval time.

***BATCHINT**
Batch interval in milliseconds.

The filter value is the integer interval time.

***BATCHSIZE**
Batch size.

The filter value is the integer batch size.

***CLNTWGHT**
Client channel weight.

The filter value is the integer client channel weight.

***CLUSNL**
Cluster namelist.

The filter value is the list of cluster names.

***CLUSTER**
The cluster to which the channel belongs.

The filter value is the name of the cluster.

***CLWLRANK**
Cluster workload rank.

The filter value is the integer rank.

***CLWLPRTY**
Cluster workload priority.

The filter value is the integer priority.

***CLWLWGHT**
Cluster workload weight.

The filter value is the integer weight.

***COMPHDR**
Header compression.

The filter value is one of the following:

***NONE**
No header data compression is performed.

***SYSTEM**
Header data compression is performed.

***COMPMSG**
Message compression.

The filter value is one of the following:

***NONE**
No message data compression is performed.

***RLE** Message data compression is performed using RLE.

***ZLIBHIGH**
Message data compression is performed using ZLIB compression. A high level of compression is preferred.

***ZLIBFAST**
Message data compression is performed using ZLIB compression. A fast compression time is preferred.

***ANY** Any compression technique supported by the queue manager can be used.

***CONNAME**
Remote connection name.

The filter value is the connection name string.

***CVTMSG**
Whether the message is converted before transmission.

The filter value is one of the following:

***YES** The application data in the message is converted before sending.

***NO** The application data in the message is not converted before sending.

***DSCITV**
Disconnect interval in seconds.

The filter value is the integer interval time.

***HRTBTINTVL**
Heartbeat interval in seconds.

The filter value is the integer interval time.

***KAINT**
Keep alive interval in seconds.

The filter value is the integer interval time.

***LOCLADDR**
Local connection name.

The filter value is the connection name string.

***LONGRTY**
Long retry count.

The filter value is the integer count.

***LONGTMR**
Long retry interval in seconds.

The filter value is the integer interval time.

***MAXINST**
Maximum instances of an individual server-connection channel.

The filter value is the integer number of instances.

***MAXINSTC**
Maximum instances of an individual server-connection channel from a single client.

The filter value is the integer number of instances.

***MAXMSGLEN**
Maximum message length.

The filter value is the integer length.

***MCANAME**
Message channel agent name.

The filter value is the agent name.

***MCATYPE**
Whether the message channel agent program should run as a thread or process.

The filter value is one of the following:

***PROCESS**
The message channel agent runs as a separate process.

***THREAD**
The message channel agent runs as a separate thread.

***MCAUSRID**
Message channel agent user identifier.

The filter value is the user identifier string.

***MODENAME**
SNA mode name.

The filter value is the mode name string.

***MONCHL**
Channel Monitoring.

The filter value is one of the following:

***QMGR**
The collection of Online Monitoring Data is inherited from the setting of the queue manager attribute MONCHL.

***OFF** Online Monitoring Data collection for this channel is switched off.

***LOW** Monitoring data collection is turned on with a low ratio of data collection.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection.

***MSGEXIT**
Message exit name.

The filter value is the exit name.

***MSGRTYDATA**
Message retry exit user data.

The filter value is the user data string.

***MSGRTYEXIT**
Message retry exit name.

The filter value is the exit name.

***MSGRTYITV**
Message retry interval interval in seconds.

The filter value is the integer interval time.

***MSGRTYNBR**
Number of message retries.

The filter value is the integer number of retries.

***MSGUSRDATA**
Message exit user data.

The filter value is the user data string.

***NETPRTY**
Network connection priority ranging from 0 through 9.

The filter value is the integer priority value.

***NPMSPEED**
Whether the channel supports fast nonpersistent messages.

The filter value is one of the following:

***FAST** The channel supports fast nonpersistent messages.

***NORMAL**
The channel does not support fast nonpersistent messages.

***PROPCTL**
Message Property Control.

The filter value is one of the following:

***COMPAT**
Compatibility mode.

***NONE**
No properties sent to remote queue manager.

***ALL** All properties sent to remote queue manager.

***PUTAUT**
Whether the user identifier in the context information is used.

The filter value is one of the following:

***DFT** No authority check is made before the message is put on the destination queue.

***CTX** The user identifier in the message context information is used to establish authority to put the message.

***RCVEXIT**
Receive exit name.

The filter value is the exit name.

***RCVUSRDATA**
Receive exit user data.

The filter value is the user data string.

***SCYEXIT**
Security exit name.

The filter value is the exit name.

***SCYUSRDATA**
Security exit user data.

The filter value is the user data string.

***SEQNUMWRAP**
Maximum message sequence number.

The filter value is the integer sequence number.

***SHARECNV**
The number of shared conversations over a TCP/IP socket.

The filter value is the integer number of shared conversations.

***SHORTRTY**
Short retry count.

The filter value is the integer count.

***SHORTTMR**
Short retry interval in seconds.

The filter value is the integer interval time.

***SNDEXIT**
Send exit name.

The filter value is the exit name.

***SNDUSRDATA**
Send exit user data.

The filter value is the user data string.

***SSLCAUTH**
Whether the channel should carry out client authentication over SSL.

The filter value is one of the following:

***REQUIRED**
Client authentication is required.

***OPTIONAL**
Client authentication is optional.

***SSLCIPH**
The CipherSpec using in SSL channel negotiation.

The filter value is the name of the CipherSpec.

***SSLPEER**
The X500 peer name used in SSL channel negotiation.

The filter value is the peer name.

***STATCHL**
Channel Statistics.

The filter value is one of the following:

***QMGR**
The collection of statistics data is inherited from the setting of the queue manager attribute STATCHL.

***OFF** Statistics data collection for this channel is switched off.

***LOW** Statistics data collection is turned on with a low ratio of data collection.

***MEDIUM**
Statistics data collection is turned on with a moderate ratio of data collection.

***HIGH**
Statistics data collection is turned on with a high ratio of data collection.

***TEXT**
Descriptive comment.

The filter value is the text description of the channel.

***TGTMQMNAME**
Target queue manager name.

The filter value is the target queue manager of the channel.

***TMQNAME**
Transmission queue name.

The filter value is the name of the queue.

***TPNAME**
The SNA transaction program name.

The filter value is the program name string.

***TRPTYPE**
Transport type.

The filter value is one of the following:

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

***LU62** SNA LU 6.2.

***USERID**
Task user identifier.

The filter value is the user identifier string.

Examples

None

Error messages

Unknown

Work with MQ Channel Status (WRKMQMCHST)

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

The Work with MQ Channel Status (WRKMQMCHST) command allows you to work with the status of one or more channel definitions.

Parameters

Keyword	Description	Choices	Notes
CHLNAME	Channel name	*Character value*, ***ALL**	Optional, Positional 1
CONNAME	Connection name	*Character value*, ***ALL**	Optional, Positional 2
TMQNAME	Transmission queue name	*Character value*, ***ALL**	Optional, Positional 3
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 4
CHLSTS	Channel status	***ALL**, *SAVED, *CURRENT	Optional, Positional 5

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 6
	Element 1: Filter keyword	*CHLSTS, *CHLTYPE, *COMPHDR, *COMPMSG, *CONNAME, *INDOUBT, *INDMSGS, *INDSEQNO, *LSTSEQNO, *MONCHL, *RMTMQMNAME, *SHARECNV, *STATUS, *SUBSTATE, *TMQNAME, *XQMSGSA	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Channel name (CHLNAME)

Specifies the name of the channel definition.

The possible values are:

***ALL** All channel definitions are selected.

generic-channel-name
Specify the generic name of the channel definitions to be selected. A generic name is a character string followed by an asterisk (*). For example ABC*, it selects all channel definitions having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

channel-name
Specify the name of the channel definition.

Connection name (CONNAME)

Specifies the name of the machine to connect.

The possible values are:

***ALL** All the channels are selected.

generic-connection-name
Specify the generic connection name of the required channels.

connection-name
Specify the connection name of the required channels.

Transmission queue name (TMQNAME)

Specifies the name of the transmission queue.

The possible values are:

***ALL** All the transmission queues are selected.

generic-transmission-queue-name
Specify the generic name of the transmission queues.

transmission-queue-name
Specify the name of the transmission queue. A transmission queue name is required if the channel definition type (CHLTYPE) is *SDR or *SVR.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used. If you do not have a default queue manager defined on the system, the command fails.

message-queue-manager-name
The name of a message queue manager.

Channel status (CHLSTS)

Specifies the type of channel status to display.

The possible values are:

***SAVED**
Saved channel status only is displayed. Status is not saved until a persistent message is transmitted across a channel, or a nonpersistent message is transmitted with a NPMSPEED of NORMAL. Because status is saved at the end of each batch, a channel has no saved status until at least one batch has been transmitted.

***CURRENT**
Current channel status only is displayed. This applies to channels that have been started, or on which a client has connected, and that have not finished or disconnected normally. The current status data is updated as messages are sent or received.

***ALL** Both saved and current channel status is displayed.

Filter command (WHERE)

This parameter can be used to selectively display the status of only those channels with particular channel status attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***CHLSTS**
The type of channel status.

The filter value is one of the following:

***CURRENT**
Current status for an active channel.

***SAVED**
Saved status for an active or inactive channel.

***CHLTYPE**
The type of channel.

The filter value is one of the following:

***SDR** Sender channel.

***SVR** Server channel.

***RCVR**
Receiver channel.

***RQSTR**
Requester channel.

***CLUSSDR**
Cluster-sender channel.

***CLUSRCVR**
Cluster-receiver channel.

***SVRCN**
Server-connection channel.

***COMPHDR**
Whether the channel performs header data compression.

The filter value is one of the following:

***NONE**
No header data compression is performed.

***SYSTEM**
Header data compression is performed.

***COMPMSG**
Whether the channel performs message data compression.

The filter value is one of the following:

***NONE**
No message data compression is performed.

***RLE** Message data compression is performed using RLE.

***ZLIBHIGH**
Message data compression is performed using ZLIB compression. A high level of compression is preferred.

***ZLIBFAST**
Message data compression is performed using ZLIB compression. A fast compression time is preferred.

***CONNAME**
The connection name of the channel.

The filter value is the connection name string.

***INDOUBT**
Whether there are any in-doubt messages in the network.

The filter value is either *NO or *YES.

***INDMSGS**
The number of in-doubt message.

The filter value is the integer number of messages.

***INDSEQNO**
The sequence number of the message that is in-doubt.

The filter value is the integer sequence number.

***LSTSEQNO**
The last message sequence number.

The filter value is the integer sequence number.

***MONCHL**
The current level of monitoring data collection for the channel.

The filter value is one of the following:

***NONE**
No monitoring data is collected.

***LOW** A low ratio of monitoring data is collected.

***MEDIUM**
A medium ratio of monitoring data is collected.

***HIGH**
A high ratio of monitoring data is collected.

***RMTMQMNAME**
The remote message queue manager.

The filter value is the message queue manager name.

***SHARECNV**
The number of shared conversations over a TCP/IP socket.

The filter value is the integer number of shared conversations.

***STATUS**
The status of the channel.

The filter value is one of the following:

***STARTING**
The channel is ready to beging negotiation with the target MCA.

***BINDING**
The channel is establishing a session.

***INACTIVE**
The channel has ended processing normally or the channel has never started.

***INITIALIZING**
The channel initiator is attempting to start the channel.

***RUNNING**
The channel is transferring or is ready to transfer data.

***STOPPING**
The channel has been requested to stop.

***RETRYING**
A previous attempt to establish a connection has failed. The channel will retry the connection after the specified interval.

***PAUSED**
The channel is waiting for the message retry interval.

***STOPPED**
The channel has been stopped.

***REQUESTING**
The channel has been requested to start.

***SUBSTATE**
The channel substate.

The filter value is one of the following:

***ENDBATCH**
End of batch processing.

***SEND**
Sending data.

***RECEIVE**
Receiving data.

***SERIALIZE**
Serializing with the partner channel.

***RESYNCH**
Re-synchronizing with the partner channel.

***HEARTBEAT**
Heartbeat processing.

***SCYEXIT**
Processing a security exit.

***RCVEXIT**
Processing a receive exit.

***SENDEXIT**
Processing a send exit.

***MSGEXIT**
Processing a message exit.

***MREXIT**
Processing a message-retry exit.

***CHADEXIT**
Processing a channel auto-definition exit.

***NETCONNECT**
Connecting to remote machine.

***SSLHANDSHK**
Establishing an SSL connection.

***NAMESERVER**
Requesting information from a name server.

***MQPUT**
MQPUT processing.

***MQGET**
MQGET processing.

***MQICALL**
Processing an MQI call.

***COMPRESS**
Compressing or decompressing data.

***TMQNAME**
The transmission queue of the channel.

The filter value is the queue name.

***XQMSGSA**
The number of messages queued on the transmission queue available for MQGET. This field is valid for cluster-sender channels.

The filter value is the integer number of messages.

Examples

None

Error messages

Unknown

Work with MQ Clusters (WRKMQMCL)

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

The Work with MQ Clusters (WRKMQMCL) command allows you to work with multiple cluster-queue-manager definitions that are defined on the local queue manager.

Parameters

Keyword	Description	Choices	Notes
CLUSQMGR	Cluster Queue Manager name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *ALTTIME, *BATCHHB, *BATCHINT, *BATCHSIZE, *CHLNAME, *CLUSDATE, *CLUSQMGR, *CLUSTER, *CLUSTIME, *CLWLPRTY, *CLWLRANK, *CLWLWGHT, *COMPHDR, *COMPMSG, *CONNAME, *CVTMSG, *DFNTYPE, *DSCITV, *HRTBTINTVL, *KAINT, *LOCLADDR, *LONGRTY, *LONGTMR, *MAXMSGLEN, *MCANAME, *MCATYPE, *MCAUSRID, *MONCHL, *MSGEXIT, *MSGRTYDATA, *MSGRTYEXIT, *MSGRTYITV, *MSGRTYNBR, *MSGUSRDATA, *NETPRTY, *NPMSPEED, *PUTAUT, *QMID, *QMTYPE, *RCVEXIT, *RCVUSRDATA, *SCYEXIT, *SCYUSRDATA, *SEQNUMWRAP, *SHORTRTY, *SHORTTMR, *SNDEXIT, *SNDUSRDATA, *SSLCAUTH, *SSLCIPH, *SSLPEER, *STATCHL, *STATUS, *SUSPEND, *TEXT, *TRPTYPE, *USERID	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Cluster Queue Manager name (CLUSQMGR)

Specifies the name or names of the cluster-queue-manager definitions.

***ALL** All cluster-queue-manager definitions are selected.

generic-cluster-queue-manager-name
Specify the generic name of the MQ cluster-queue-manager definitions. A generic name is a character string followed by an asterisk (*)> For example ABC*, it selects all cluster-queue-manager definitions having names that start with the character string. You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered. You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

cluster-queue-manager-name
Specify the name of the MQ cluster-queue-manager definition.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Filter command (WHERE)

This parameter can be used to selectively display only those cluster-queue-managers with particular attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***ALTDATE**
The date on which the definition or information was last altered.

The filter value is the data in the form yyyy-mm-dd.

***ALTTIME**
The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

***BATCHHB**
Batch heartbeat interval in milliseconds.

The filter value is the integer interval time.

***BATCHINT**
Batch interval in milliseconds.

The filter value is the integer interval time.

***BATCHSIZE**
Batch size.

The filter value is the integer batch size.

***CHANNEL**
The channel name of the cluster-queue-manager.

The filter value is the name of the channel.

***CLUSDATE**
The date on which the definition became available to the local queue manager.

The filter value is the data in the form yyyy-mm-dd.

***CLUSQMGR**
The cluster-queue-manager name.

The filter value is the name of the cluster-queue-manager.

***CLUSTER**
The cluster to which the cluster-queue-manager belongs.

The filter value is the name of the cluster.

***CLUSTIME**
The time at which the definition became available to the local queue manager.

The filter value is the time in the form hh:mm:ss.

***CLWLRANK**
Cluster workload rank.

The filter value is the integer rank.

***CLWLPRTY**
Cluster workload priority.

The filter value is the integer priority.

***CLWLWGHT**
Cluster workload weight.

The filter value is the integer weight.

***COMPHDR**
Header compression.

The filter value is one of the following:

***NONE**
No header data compression is performed.

***SYSTEM**
Header data compression is performed.

***COMPMSG**
Message compression.

The filter value is one of the following:

***NONE**
No message data compression is performed.

***RLE** Message data compression is performed using RLE.

***ZLIBHIGH**
Message data compression is performed using ZLIB compression. A high level of compression is preferred.

***ZLIBFAST**
Message data compression is performed using ZLIB compression. A fast compression time is preferred.

***ANY** Any compression technique supported by the queue manager can be used.

***CONNAME**
Remote connection name.

The filter value is the connection name string.

***CVTMSG**
Whether the message should be converted before transmission.

The filter value is one of the following:

***YES** The application data in the message is converted before sending.

***NO** The application data in the message is not converted before sending.

***DFNTYPE**
How the cluster channel was defined.

The filter value is one of the following:

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

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

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

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

***DSCITV**
Disconnect interval in seconds.

The filter value is the integer interval time.

***HRTBTINTVL**
Heartbeat interval in seconds.

The filter value is the integer interval time.

***KAINT**
Keep alive interval in seconds.

The filter value is the integer interval time.

***LOCLADDR**
Local connection name.

The filter value is the connection name string.

***LONGRTY**
Long retry count.

The filter value is the integer count.

***LONGTMR**
Long retry interval in seconds.

The filter value is the integer interval time.

***MAXMSGLEN**
Maximum message length.

The filter value is the integer length.

***MCANAME**
Message channel agent name.

The filter value is the agent name.

***MCATYPE**
Whether the message channel agent program should run as a thread or process.

The filter value is one of the following:

***PROCESS**
The message channel agent runs as a separate process.

***THREAD**
The message channel agent runs as a separate thread.

***MCAUSRID**
Message channel agent user identifier.

The filter value is the user identifier string.

***MONCHL**
Channel Monitoring.

The filter value is one of the following:

***QMGR**
The collection of Online Monitoring Data is inherited from the setting of the queue manager attribute MONCHL.

***OFF** Online Monitoring Data collection for this channel is switched off.

***LOW** Monitoring data collection is turned on with a low ratio of data collection.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection.

***MSGEXIT**
Message exit name.

The filter value is the exit name.

***MSGRTYDATA**
Message retry exit user data.

The filter value is the user data string.

***MSGRTYEXIT**
Message retry exit name.

The filter value is the exit name.

***MSGRTYITV**
Message retry interval interval in seconds.

The filter value is the integer interval time.

***MSGRTYNBR**
Number of message retries.

The filter value is the integer number of retries.

***MSGUSRDATA**
Message exit user data.

The filter value is the user data string.

***NETPRTY**
Network connection priority in the range 0 through 9.

The filter value is the integer priority value.

***NPMSPEED**
Whether the channel supports fast non persistent messages.

The filter value is one of the following:

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

***NORMAL**
The channel does not support fast non persistent messages.

***PUTAUT**
Whether the user identifier in the context information should be used.

The filter value is one of the following:

***DFT** No authority check is made before the message is put on the destination queue.

***CTX** The user identifier in the message context information is used to establish authority to put the message.

***QMID**
The internally generated unique name of the cluster-queue-manager.

The filter value is the unique name.

***QMTYPE**
The function of the cluster-queue-manager in the cluster.

The filter value is one of the following:

***REPOS**
Provides a full repository service.

***NORMAL**
Does not provide a full repository service.

***RCVEXIT**
Receive exit name.

The filter value is the exit name.

***RCVUSRDATA**
Receive exit user data.

The filter value is the user data string.

***SCYEXIT**
Security exit name.

The filter value is the exit name.

***SCYUSRDATA**
Security exit user data.

The filter value is the user data string.

***SEQNUMWRAP**
Maximum message sequence number.

The filter value is the integer sequence number.

***SHORTRTY**
Short retry count.

The filter value is the integer count.

***SHORTTMR**
short retry interval in seconds.

The filter value is the integer interval time.

***SNDEXIT**
Send exit name.

The filter value is the exit name.

***SNDUSRDATA**
Send exit user data.

The filter value is the user data string.

***SSLCAUTH**
Whether the channel should carry out client authentication over SSL.

The filter value is one of the following:

***REQUIRED**
Client authentication is required.

***OPTIONAL**
Client authentication is optional.

***SSLCIPH**
The CipherSpec using in SSL channel negotiation.

The filter value is the name of the CipherSpec.

***SSLPEER**
The X500 peer name used in SSL channel negotiation.

The filter value is the peer name.

***STATCHL**
Channel Statistics.

The filter value is one of the following:

***QMGR**
The collection of statistics data is inherited from the setting of the queue manager attribute STATCHL.

***OFF** Statistics data collection for this channel is switched off.

***LOW** Statistics data collection is turned on with a low ratio of data collection.

***MEDIUM**
Statistics data collection is turned on with a moderate ratio of data collection.

***HIGH**
Statistics data collection is turned on with a high ratio of data collection.

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

The filter value is one of the following:

***STARTING**
The channel is waiting to become active.

***BINDING**
The channel is performing channel negotiation.

***INACTIVE**
The channel is not active.

***INITIALIZING**
The channel initiator is attempting to start a channel.

***RUNNING**
The channel is either transferring messages, or is waiting for messages to arrive on the transmission queue.

***STOPPING**
The channel is stopping, or a close request has been received.

***RETRYING**
A previous attempt to establish a connection has failed. The MCA will reattempt connection after the specified time interval.

***PAUSED**
The channel is waiting for the message-retry interval to complete before retrying an MQPUT operation.

***STOPPED**
The channel has either been manually stopped, or the retry limit has been reached.

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

***SUSPEND**
Whether this cluster queue manager is suspended from the cluster or not.

The filter value is either *NO or *YES.

***TEXT**
Descriptive comment.

The filter value is the text description of the channel.

***TMQNAME**
Transmission queue name.

The filter value is the name of the queue.

***USERID**
Task user identifier.

The filter value is the user identifier string.

Examples

None

Error messages

Unknown

Work with MQ Cluster Queues (WRKMQMCLQ)

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

The Work with MQ Cluster Queues (WRKMQMCLQ) command allows you to work with cluster queues that are defined on the local queue manager.

Parameters

Keyword	Description	Choices	Notes
QNAME	Queue name	*Character value*, ***ALL**	Optional, Positional 1

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
CLUSTER	Cluster name	*Character value*, ***ALL**	Optional, Positional 3
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 4
	Element 1: Filter keyword	*ALTDATE, *ALTTIME, *CLUSDATE, *CLUSQMGR, *CLUSQTYPE, *CLUSTER, *CLUSTIME, *DEFBIND, *DFTMSGPST, *DFTPTY, *PUTENBL, *QMID, *TEXT	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Queue name (QNAME)

Specifies the name or names of the cluster queue definitions.

***ALL** All cluster queue definitions are selected.

generic-queue-name
Specify the generic name of the MQ cluster-queue definitions. A generic name is a character string followed by an asterisk (*). For example ABC*, it selects all cluster-queue definitions having names that start with the character string. You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered. You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

queue-name
Specify the name of the MQ cluster-queue definition.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

***DFT** Use the default queue manager.

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

Cluster name (CLUSTER)

Specifies the name of the cluster.

***ALL** All cluster definitions are selected.

generic-cluster-name
Specify the generic name of the MQ cluster definitions. A generic name is a character string followed by an asterisk (*). For example ABC*, it selects all cluster definitions having names that start with the character string. You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered. You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

cluster-name
Specify the name of the MQ cluster definition.

Filter command (WHERE)

This parameter can be used to selectively display only those cluster queues with particular cluster queue attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***ALTDATE**
The date on which the definition or information was last altered.

The filter value is the data in the form yyyy-mm-dd.

***ALTTIME**
The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

***CLUSDATE**
The date on which the definition became available to the local queue manager.

The filter value is the date in the form yyyy-mm-dd.

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

The filter value is the name of the queue manager.

***CLUSQTYPE**
Cluster queue type.

The filter value is one of the following:

***LCL** The cluster queue represents a local queue.

***ALS** The cluster queue represents an alias queue.

***RMT** The cluster queue represents a remote queue.

***MQMALS**
The cluster queue represents a queue manager alias.

***CLUSTER**
The name of the cluster that the queue is in.

The filter value is the name of the cluster.

***CLUSTIME**
The time at which the definition became available to the local queue manager.

The filter value is the time in the form hh:mm:ss.

***DEFBIND**
Default message binding.

The filter value is one of the following:

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

***NOTFIXED**
The queue handle is not bound to any particular instance of the cluster queue.

***DFTMSGPST**
Default persistence of the messages put on this queue.

The filter value is one of the following:

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

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

***DFTPTY**
Default priority of the messages put on the queue.

The filter value is the integer priority value.

***PUTENBL**
Whether applications are permitted to put messages to the queue.

The filter value is one of the following:

***NO** Messages cannot be added to the queue.

***YES** Messages can be added to the queue by authorized applications.

***QMID**
Internally generated unique name of the queue manager that hosts the queue.

The filter value is the name of the queue manager.

***TEXT**
Descriptive comment.

The filter value is the text description of the queue.

Examples

None

Error messages

Unknown

Work with MQ Connections (WRKMQMCONN)

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

The Work with MQ Connections (WRKMQMCONN) command allows you to work with connection information for applications that are connected to the queue manager.

This enables you to display connection handles and end connections to the queue manager.

Parameters

Keyword	Description	Choices	Notes
CONN	Connection Identifier	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*APPLTAG, *APPLTYPE, *CHLNAME, *CONNAME, *PID, *TID, *UOWLOGDA, *UOWLOGTI, *UOWSTDA, *UOWSTTI, *URTYPE, *USERID	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Connection Identifier (CONN)

The connection identifiers to work with.

The possible values are:

***ALL** All connection identifiers are selected.

connection-id
Specify the name of a specific connection identifier. The connection identifier is a 16 character hex string.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Filter command (WHERE)

This parameter can be used to selectively display only those queue manager connections with particular connection attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***APPLTAG**
The tag of the application connected to the queue manager.

The filter value is the application tag string.

***APPLTYPE**
The type of applcation connected to the queue manager.

The filter value is one of the following:

***CICS** CICS/400 application.

***MVS** MVS application.

***IMS** IMS application.

***OS2** OS/2 application.

***DOS** DOS application.

***UNIX**
UNIX application.

***QMGR**
Queue manager application.

***OS400**
i5/OS application.

***WINDOWS**
Windows application.

***CICS_VSE**
CICS/VSE application.

***WINDOWS_NT**
Windows NT application.

***VMS** VMS application.

***NSK** Tandem/NSK application.

***VOS** VOS application.

***IMS_BRIDGE**
IMS bridge application.

***XCF** XCF application.

***CICS_BRIDGE**
CICS bridge application.

***NOTES_AGENT**
Lotus Notes application.

***BROKER**
Broker application.

***JAVA** Java application.

***DQM**
DQM application.

***CHINIT**
Channel initiator.

user-value
User-defined application.

The filter value is the integer application type.

***CHLNAME**
The name of the channel that owns the connection.

The filter value is the channel name.

***CONNAME**
The connection name associated with the channel that owns the connection.

The filter value is the connection name.

***PID** The process identifier of the application that is connected to the queue manager.

The filter value is the process identifier integer.

***TID** The thread identifier of the application that is connected to the queue manager.

The filter value is the thread identifier integer.

***UOWLOGDA**
The date that the transaction associated with the connection first wrote to the log.

The filter value is the date in the form yyyy-mm-dd.

***UOWLOGTI**
The time that the transaction associated with the connection first wrote to the log.

The filter value is the time in the form hh:mm:ss.

***UOWSTDA**
The date that the transaction associated with the connection was started.

The filter value is the date in the form yyyy-mm-dd.

***UOWSTTI**
The time that the transaction associated with the connection was started.

The filter value is the time in the form hh:mm:ss.

***URTYPE**
The type of unit of recovery identifier as seen by the queue manager.

The filter value is one of the following:

***QMGR**
A queue manager transaction.

***XA** An externally coordinated transaction. This includes units of work which have been established using i5/OS Start Commitment Control (STRCMTCTL).

***USERID**
The user identifier associated with the connection.

The filter value is user identifier name.

Examples

None

Error messages

Unknown

Work with MQ Listeners (WRKMQMLSR)

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

The Work with MQ Listener objects (WRKMQMLSR) command allows you to work with listener objects which are defined on the local queue manager.

This enables you to change, copy, create, delete, start, stop & display listener objects display and change authority to an MQ listener object.

This command also enables you to view the current status of all running listeners on the current system.

Parameters

Keyword	Description	Choices	Notes
OPTION	Option	***STATUS**, *OBJECT	Optional, Positional 1
LSRNAME	Listener name	*Character value*, ***ALL**	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 4
	Element 1: Filter keyword	*ALTDATE, *ALTTIME, *BACKLOG, *CONTROL, *IPADDR, *PORT, *TEXT	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Option (OPTION)

This option enables you to select whether you want to information on listener status or listener object definitions.

The possible values are:

***STATUS**
Listener status information is displayed.

***OBJECT**
Listener object information is displayed.

Listener name (LSRNAME)

The name or names of the listener objects.

The possible values are:

***ALL or ***
All listener objects are selected.

generic-listener-name
The generic name of the listener objects. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all listener objects having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

listener-name
Specify the name of a single listener object.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Filter command (WHERE)

This parameter can be used to selectively display only those listener objects with particular listener attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***ALTDATE**
The date on which the definition or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

***ALTTIME**
The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

***BACKLOG**
The number of concurrent connection requests supported.

The filter value is the integer backlog value.

***CONTROL**
Whether the listener is started and stopped with the queue manager.

The filter value is one of the following:

***MANUAL**
The listener is not automatically started or stopped.

***QMGR**
The listener is started and stopped as the queue manager is started and stopped.

***STARTONLY**
The listener is started as the queue manager is started, but is not requested to stop when the queue manager is stopped.

***IPADDR**
The local IP Address to be used by the listener.

The filter value is the IP Address.

***PORT**
The port number to be used by the listener.

The filter value is the integer port value.

***TEXT**
Descriptive comment.

The filter value is the text description of the listener.

Examples

None

Error messages

Unknown

Work with MQ Messages (WRKMQMMSG)

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

The Work with MQ Messages (WRKMQMMSG) command lists the messages on a specified local queue and allows you to work with those messages. From the list of messages, you can display the contents of a message and its associated message descriptor (MQMD).

Parameters

Keyword	Description	Choices	Notes
QNAME	Queue name	*Character value*	Required, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
FIRST	First Message	1-30000, **1**	Optional, Positional 3
MAXMSG	Maximum number of messages	1-30000, **48**	Optional, Positional 4
MAXMSGLEN	Maximum message size	128-999999, **1024**	Optional, Positional 5

Queue name (QNAME)

Specifies the name of the local queue.

The possible values are:

queue-name
Specify the name of the local queue.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

First Message (FIRST)

Specifies the number of the first message to display.

The possible values are:

1 The number of the first message to display is 1.

message-number
Specify the number of the first message to display ranging from 1 through 30 000.

Maximum number of messages (MAXMSG)

Specifies the maximum number of messages to display.

The possible values are:

48 Display a maximum of 48 messages.

count-value
Specify a value for the maximum number of messages to display ranging from 1 through 30 000.

Maximum message size (MAXMSGLEN)

Specifies the maximum size of message data to display.

The size of a message, greater than the value specified, is suffixed by a plus (+) character to indicate that the message data is truncated.

The possible values are:

1024 The size of the message data is 1024 bytes.

length-value
Specify a value ranging from 128 through 999999.

Examples

None

Error messages

Unknown

Work with MQ Namelist (WRKMQMNL)

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

The Work with MQ Namelists (WRKMQMNL) command allows you to work with multiple namelist definitions that are defined on the local queue manager. This enables you to copy, change, display, delete, display authority and edit authority of an MQ namelist object.

Parameters

Keyword	Description	Choices	Notes
NAMELIST	Namelist	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *ALTTIME, *NAMECNT, *NAMES, *TEXT	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Namelist (NAMELIST)

Specifies the name or names of the namelists.

The possible values are:

***ALL** All namelist definitions are selected.

generic-namelist-name
Specify the generic name of the MQ namelists. A generic name is a character string followed by an asterisk (*). For example ABC*, it selects all namelists having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

namelist-name
Specify the name of the MQ namelist.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** The default queue manager is used.

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

Filter command (WHERE)

This parameter can be used to selectively display only those namelists with particular namelist attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***ALTDATE**
The date on which the definition or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

***ALTTIME**
The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

***NAMECNT**
The number of names in the namelist.

The filter value is the integer number of names.

***NAMES**
The names in the namelist.

The filter value is the string name.

***TEXT**
Descriptive comment.

The filter value is the text description of the queue.

Examples

None

Error messages

Unknown

Work with MQ Processes (WRKMQMPRC)

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

The Work with MQ Processes (WRKMQMPRC) command allows you to work with multiple process definitions that are defined on the local queue manager. This enables you to copy, change, display, delete, display authority, and edit authority of an MQ process object.

Parameters

Keyword	Description	Choices	Notes
PRCNAME	Process name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *ALTTIME, *APPID, *APPTYPE, *ENVDATA, *TEXT, *USRDATA	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Process name (PRCNAME)

Specifies the name or names of the process definitions.

The possible values are:

***ALL** All process definitions are selected.

generic-process-name
Specify the generic name of the MQ process definitions. A generic name is a character string followed by an asterisk (*). For example ABC*, it selects all process definitions having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

process-name
Specify the name of the MQ process definition.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Filter command (WHERE)

This parameter can be used to selectively display only those processes with particular process attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***ALTDATE**
The date on which the definition or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

***ALTTIME**
The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

***APPID**
The name of the application to start.

The filter value is the name of the application.

***APPTYPE**
The type of the application to start.

The filter value is one of the following:

***CICS** CICS/400 application.

***MVS** MVS application.

***IMS** IMS application.

***OS2** OS/2 application.

***DOS** DOS application.

***UNIX**
UNIX application.

***QMGR**
Queue manager application.

***OS400**
i5/OS application.

***WINDOWS**
Windows application.

***CICS_VSE**
CICS/VSE application.

***WINDOWS_NT**
Windows NT application.

***VMS** VMS application.

***NSK** Tandem/NSK application.

***VOS** VOS application.

***IMS_BRIDGE**
IMS bridge application.

***XCF** XCF application.

***CICS_BRIDGE**
CICS bridge application.

***NOTES_AGENT**
Lotus Notes application.

***BROKER**
Broker application.

***JAVA** Java application.

***DQM**
DQM application.

user-value
User-defined application.

The filter value is the integer application type.

***ENVDATA**
Environment data pertaining to the application.

The filter value is the environment data.

***TEXT**
Descriptive comment.

The filter value is the text description of the queue.

***USRDATA**
User data pertaining to the application.

The filter value is the user data.

Examples

None

Error messages

Unknown

Work with MQ Queues (WRKMQMQ)

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

The Work with MQ Queues (WRKMQMQ) command allows you to work with multiple queues that are defined on the local queue manager. This enables you to copy, change, display, delete, display authority and edit authority of an MQ Queue object.

Parameters

Keyword	Description	Choices	Notes
QNAME	Queue name	*Character value*, ***ALL**	Optional, Positional 1
QTYPE	Queue type	***ALL** , *ALS, *LCL, *MDL, *RMT	Optional, Positional 2
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 3
CLUSTER	Cluster name	*Character value*, ***ALL**	Optional, Positional 4
CLUSNL	Cluster namelist name	*Character value*, ***ALL**	Optional, Positional 5
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 6
	Element 1: Filter keyword	*ACCTQ, *ALTDATE, *ALTTIME, *BKTTHLD, *BKTQNAME, *CLUSDATE, *CLUSNL, *CLUSQMGR, *CLUSQTYPE, *CLUSTER, *CLUSTIME, *CLWLPRTY, *CLWLRANK, *CLWLUSEQ, *CRDATE, *CRTIME, *CURDEPTH, *DEFBIND, *DFTPUTRESP, *DFNTYPE, *DFTMSGPST, *DFTPTY, *DFTSHARE, *DISTLIST, *FULLEVT, *GETDATE, *GETENBL, *GETTIME, *HDNBKTCNT, *HIGHEVT, *HIGHTHLD, *INITQNAME, *IPPROCS, *JOBS, *LOWEVT, *LOWTHLD, *MAXDEPTH, *MAXMSGLEN, *MEDIAREC, *MONQ, *MSGAGE, *MSGDLYSEQ, *MSGREADAHD, *NPMCLASS, *OPPROCS, *PRCNAME, *PROPCTL, *PUTDATE, *PUTENBL, *PUTTIME, *QMID, *QTYPE, *RMTMQMNAME, *RMTQNAME, *RTNITV, *SHARE, *SRVEVT, *SRVITV, *STATQ, *TARGTYPE, *TEXT, *TGTQNAME, *TMQNAME, *TRGDATA, *TRGDEPTH, *TRGENBL, *TRGMSGPTY, *TRGTYPE, *UNCOM, *USAGE	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Queue name (QNAME)

The name or names of the queues to be selected. The queues selected by this parameter can be further limited to a particular type, if the QTYPE keyword is specified.

The possible values are:

***ALL** All queues are selected.

generic-queue-name
Specify the generic name of the queues to be selected. A generic name is a character string, followed by an asterisk (*). For example ABC*, it selects all queues having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

queue-name
Specify the name of the queue.

Queue type (QTYPE)

This parameter can be specified to limit the queues that are displayed to a particular type.

The possible values are:

***ALL** All queue types.

***ALS** Alias queues.

***LCL** Local queues.

***MDL** Model queues.

***RMT** Remote queues.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Cluster name (CLUSTER)

This parameter can be specified to limit the queues that are displayed to be members of a particular cluster.

The possible values are:

***ALL** All clusters.

generic-cluster-name
The generic name of a cluster.

cluster-name
The name of a cluster.

Cluster namelist name (CLUSNL)

This parameter can be specified to limit the queues that are displayed to be members of clusters within a cluster namelist.

The possible values are:

***ALL** All cluster namelists.

generic-cluster-namelist-name
The generic name of a cluster namelist.

cluster-namelist-name
The name of a cluster namelist.

Filter command (WHERE)

This parameter can be used to selectively display only those queues with particular queue attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***ACCTQ**
Queue Accounting.

The filter value is one of the following:

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

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

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

***ALTDATE**
The date on which the definition or information was last altered.

The filter value is the data in the form yyyy-mm-dd.

***ALTTIME**
The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

***BKTTHLD**
Backout threshold.

The filter value is the integer threshold value.

***BKTQNAME**
Backout requeue name.

The filter value is the name of the queue.

***CLUSDATE**
The date on which the definition became available to the local queue manager.

The filter value is the date in the form yyyy-mm-dd.

***CLUSNL**
The namelist that defines the clusters that the queue is in.

The filter value is the name of the namelist.

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

The filter value is the name of the queue manager.

***CLUSQTYPE**
Cluster queue type.

The filter value is one of the following:

***LCL** The cluster queue represents a local queue.

***ALS** The cluster queue represents an alias queue.

***RMT** The cluster queue represents a remote queue.

***MQMALS**
The cluster queue represents a queue manager alias.

***CLUSTER**
The name of the cluster that the queue is in.

The filter value is the name of the cluster.

***CLUSTIME**
The time at which the definition became available to the local queue manager.

The filter value is the time in the form hh:mm:ss.

***CLWLPRTY**
Cluster workload priority.

The filter value is the integer priority.

***CLWLRANK**
Cluster workload rank.

The filter value is the integer rank.

***CLWLUSEQ**
Cluster workload queue use.

The filter value is one of the following:

***QMGR**
The value is inherited from the Queue Manager CLWLUSEQ attribute.

***LOCAL**
The local queue will be the sole target of the MQPUT.

***ANY** The queue manager will treat such a local queue as another instance of the cluster queue for the purposes of workload distribution.

***CRDATE**
The date on which the queue was created.

The filter value is the date in the form yyyy-mm-dd.

***CRTIME**
The time at which the queue was created.

The filter value is the time in the form hh:mm:ss.

***CURDEPTH**
Current depth of queue.

The filter value is the integer depth value.

***DEFBIND**
Default message binding.

The filter value is one of the following:

***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.

***DFTPUTRESP**
Default Put Response.

The filter value is one of the following:

***SYNC**
The put operation is issued synchronously.

***ASYNC**
The put operation is issued asynchronously.

***DFNTYPE**
Queue definition type.

The filter value is one of the following:

***PREDEF**
Predefined queue.

***PERMDYN**
Permanent dynamic queue.

***TEMPDYN**
Temporary dynamic queue.

***DFTMSGPST**
Default persistence of the messages put on this queue.

The filter value is one of the following:

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

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

***DFTPTY**
Default priority of the messages put on the queue.

The filter value is the integer priority value.

***DFTSHARE**
Default share option on a queue opened for input.

The filter value is one of the following:

***NO** The open request is for exclusive input from the queue.

***YES** The open request is for shared input from the queue.

***DISTLIST**
Whether distribution lists are supported by the partner queue manager.

The filter value is one of the following:

***NO** Distribution lists are not supported by the partner queue manager.

***YES** Distribution lists are supported by the partner queue manager.

***FULLEVT**
Whether Queue Depth Full events are generated.

The filter value is one of the following:

***NO** Queue Depth Full events are not generated.

***YES** Queue Depth Full events are generated.

***GETDATE**
The date on which the last message was got from the queue since queue manager start. This field is only present when Queue Monitoring is not set to *OFF.

The filter value is the data in the form yyyy-mm-dd.

***GETENBL**
Whether applications are permitted to get messages from the queue.

The filter value is one of the following:

***NO** Applications cannot retrieve messages from the queue.

***YES** Authorized applications can retrieve messages from the queue.

***GETTIME**
The time at which the last message was got from the queue since queue manager start. This field is only present when Queue Monitoring is not set to *OFF.

The filter value is the time in the form hh:mm:ss.

***HDNBKTCNT**
Whether the backout count is hardened.

The filter value is one of the following:

***NO** The backout count is not hardened.

***YES** The backout count is hardened.

***HIGHEVT**
Whether Queue Depth High events are generated.

The filter value is one of the following:

***NO** Queue Depth High events are not generated.

***YES** Queue Depth High events are generated.

***HIGHTHLD**
Queue Depth High event generation threshold.

The filter value is the integer threshold value.

***INITQNAME**
Initiation queue.

The filter value is the name of the queue.

***IPPROCS**
Number of handles indicating that the queue is open for input.

The filter value is the integer number of handles.

***JOBS** The current number of jobs that have the queue open.

The filter value is the integer number of jobs.

***LOWEVT**
Whether Queue Depth Low events are generated.

The filter value is one of the following:

***NO** Queue Depth Low events are not generated.

***YES** Queue Depth Low events are generated.

***LOWTHLD**
Queue Depth Low event generation threshold.

The filter value is the integer threshold value.

***MAXDEPTH**
Maximum depth of queue.

The filter value is the integer number of messages.

***MAXMSGLEN**
Maximum message length.

The filter value is the integer message length.

***MEDIAREC**
The journal receiver containing the last media recovery image. This field is only present for local queues.

The filter value is the journal receiver string.

***MONQ**
Online Monitoring Data.

The filter value is one of the following:

***QMGR**
The collection of Online Monitoring Data is inherited from the setting of the queue manager attribute MONQ.

***OFF** Online Monitoring Data collection for this queue is switched off.

***LOW** Monitoring data collection is turned on with a low ratio of data collection.

***MEDIUM**
Monitoring data collection is turned on with a moderate ratio of data collection.

***HIGH**
Monitoring data collection is turned on with a high ratio of data collection.

***MSGAGE**
The age in seconds of the oldest message on the Queue. This field is only present when Queue Monitoring is not set to *OFF.

The filter value is the integer message age.

***MSGDLYSEQ**
Message delivery sequence.

The filter value is one of the following:

***PTY** Messages are delivered in FIFO order within priority.

***FIFO** Messages are delivered in FIFO order regardless of priority.

***NPMCLASS**
Non-persistent message class.

The filter value is one of the following:

***NORMAL**
Non-persistent message class is normal.

***HIGH**
Non-persistent message class is high.

***MSGREADAHD**
Message read ahead.

The filter value is one of the following:

***DISABLED**
Read ahead is disabled.

***NO** Non-persistent messages are not sent to the client ahead of an application requesting them.

***YES** Non-persistent messages are sent to the client ahead of an application requesting them.

***OPPROCS**
Number of handles indicating that the queue is open for output.

The filter value is the integer number of handles.

***PRCNAME**
Process name.

The filter value is the name of the process.

***PROPCTL**
Message Property Control.

The filter value is one of the following:

***COMPAT**
Compatibility mode.

***NONE**
No properties are returned to the application.

***ALL** All properties are returned to the application.

***FORCE**
Properties are returned to the application in one or more MQRFH2 headers.

***PUTDATE**
The date on which the last message was put to the queue since queue manager start. This field is only present when Queue Monitoring is not set to *OFF.

The filter value is the data in the form yyyy-mm-dd.

***PUTENBL**
Whether applications are permitted to put messages to the queue.

The filter value is one of the following:

***NO** Messages cannot be added to the queue.

***YES** Messages can be added to the queue by authorized applications.

***PUTTIME**
The time at which the last message was put to the queue since queue manager start. This field is only present when Queue Monitoring is not set to *OFF.

The filter value is the time in the form hh:mm:ss.

***QMID**
Internally generated unique name of the queue manager that hosts the queue.

The filter value is the name of the queue manager.

***QTYPE**
Queue type.

The filter value is one of the following:

***LCL** Local queue.

***ALS** Alias queue.

***RMT** Remote queue.

***MDL** Model queue.

***RMTMQMNAME**
Remote queue manager name.

The filter value is the name of the queue manager.

***RMTQNAME**
Name of the local queue, as known by the remote queue manager.

The filter value is the name of the queue.

***RTNITV**
Retention interval.

The filter value is the integer interval value.

***SHARE**
Whether the queue can be shared.

The filter value is one of the following:

***NO** Only a single application instance can open the queue for input.

***YES** More than one application instance can open the queue for input.

***SRVEVT**
Whether service interval events are generated.

The filter value is one of the following:

***HIGH**
Service Interval High events are generated.

***OK** Service Interval OK events are generated.

***NONE**
No service interval events are generated.

***SRVITV**
Service interval event generation threshold.

The filter value is the integer threshold value.

***STATQ**
Statistics data.

The filter value is one of the following:

***QMGR**
Statistics data collection is based upon the setting of the queue manager attribute STATQ.

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

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

***TARGTYPE**
Target Type.

The filter value is one of the following:

***QUEUE**
Queue object.

***TOPIC**
Topic object.

***TEXT**
Descriptive comment.

The filter value is the text description of the queue.

***TGTQNAME**
Target queue for which this queue is an alias.

The filter value is the name of the queue.

***TMQNAME**
Transmission queue name.

The filter value is the name of the queue.

***TRGDATA**
Trigger data.

The filter value is the text of the trigger message.

***TRGDEPTH**
Trigger depth.

The filter value is the integer number of messages.

***TRGENBL**
Whether triggering is enabled.

The filter value is one of the following:

***NO** Triggering is not enabled.

***YES** Triggering is enabled.

***TRGMSGPTY**
Threshold message priority for triggers.

The filter value is the integer priority value.

***TRGTYPE**
Trigger type.

The filter value is one of the following:

***FIRST**
When the number of messages on the queue goes from 0 to 1.

***ALL** Every time a message arrives on the queue.

***DEPTH**
When the number of messages on the queue equals the value of the TRGDEPTH attribute.

***NONE**
No trigger messages are written.

***UNCOM**
Whether there are any uncommitted changes pending for the queue.

The filter value is one of the following:

***NO** There are no uncommitted changes pending.

***YES** There are uncommitted changes pending.

***USAGE**
Whether the queue is a transmission queue.

The filter value is one of the following:

***NORMAL**
The queue is not a transmission queue.

***TMQ** The queue is a transmission queue.

Examples

None

Error messages

Unknown

Work with Queue Status (WRKMQMQSTS)

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

The Work with Queue Status (WRKMQMQSTS) command lists the jobs which have a WebSphere MQ queue currently open. The command allows you to determine what options a queue was opened with and also allows you to check to see which channels and connections have a queue open.

Parameters

Keyword	Description	Choices	Notes
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 1
QNAME	Queue name	*Character value*	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*APPLTAG, *BROWSE, *CHLNAME, *CONNAME, *INPUT, *INQUIRE, *JOB, *OUTPUT, *SET, *URTYPE	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Queue name (QNAME)

Specifies the name of the local queue.

The possible values are:

queue-name
Specify the name of the local queue.

Filter command (WHERE)

This parameter can be used to selectively display only the jobs with particular attributes that have the queue open.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***APPLTAG**
The tag of the application which has the queue open.

The filter value is the application tag string.

***BROWSE**
Whether the job has the queue open for browsing.

The filter value is either *NO or *YES.

***CHLNAME**
The name of the channel which has the queue open.

The filter value is the channel name.

***CONNAME**
The connection name of the channel which has the queue open.

The filter value is the connection name.

***INPUT**
Whether the job has the queue open for input.

The filter value is one of the following:

***NO** The job does not have the queue open for input.

***SHARED**
The job has the queue open for shared input.

***EXCL**
The job has the queue open for exclusive input.

***INQUIRE**
Whether the job has the queue open for inquiry.

The filter value is either *NO or *YES.

***JOB** The name of the job which has the queue open.

The filter value is the job name.

***OUTPUT**
Whether the job has the queue open for output.

The filter value is either *NO or *YES.

***SET** Whether the job has the queue open for set.

The filter value is either *NO or *YES.

***URTYPE**
The type of unit of work recovery identifier.

The filter value is one of the following:

***QMGR**
Queue manager unit of work recovery identifier.

***XA** XA unit of work recovery identifier.

Examples

None

Error messages

Unknown

Work with MQ Subscriptions (WRKMQMSUB)

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

The Work with MQ Subscriptions (WRKMQMSUB) command allows you to work with multiple subscriptions that are defined on the local queue manager. This enables you to copy, change, display, and delete an MQ subscription.

Parameters

Keyword	Description	Choices	Notes
SUBNAME	Subscription name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*DEST, *DESTCLASS, *DESTCRRLID, *DESTMQM, *EXPIRY, *PSPROP, *PUBACCT, *PUBAPPID, *PUBPTY, *REQONLY, *SELECTOR, *SUBSCOPE, *SUBID, *TOPICOBJ, *TOPICSTR, *USERDATA, *VARUSER, *WSCHEMA	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Subscription name (SUBNAME)

Specifies the name or names of the subscriptions.

The possible values are:

***ALL** All subscriptions are selected.

generic-subscription-name
Specify the generic name of the MQ subscriptions. A generic name is a character string followed by an asterisk (*). For example ABC*, it selects all subscriptions having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

subscription-name
Specify the name of the MQ subscription.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Filter command (WHERE)

This parameter can be used to selectively display only those subscriptions with particular subscription attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***DEST**
The destination queue for messages published to this subscription.

The filter value is the name of the queue.

***DESTCLASS**
Specifies whether this is a managed subscription.

The filter value is one of the following:

***MANAGED**
The destination is managed.

***PROVIDED**
The destination is a queue.

***DESTCRRLID**
The correlation identifier for messages published to this subscription.

The filter value is the 48 character hexadecimal string representing the 24 byte correlation identifier.

***DESTMQM**
The destination queue manager for messages published to this subscription.

The filter value is the name of the queue manager.

***EXPIRY**
The the expiry time of the subscription.

The filter value is the integer expiry time.

***PSPROP**
The manner in which publish / subscribe related message properties are added to messages sent to this subscription.

The filter value is one of the following:

***NONE**
Publish / subscribe properties are not added to the message.

***COMPAT**
Publish / subscribe properties are added to the message to maintain compatibility with V6 Publish / Subscribe.

***RFH2**
Publish / subscribe properties are added to the message within an RFH Version 2 header.

***PUBACCT**
The accounting token for messages published to this subscription.

The filter value is the 64 character hexadecimal string representing the 32 byte publish accounting token.

***PUBAPPID**
The publish application identity for messages published to this subscription.

The filter value is the publish application identifier.

***PUBPTY**
The priority of the message sent to this subscription.

The filter value is the integer priority.

***REQONLY**
Whether the subscriber will poll for updates via MQSUBRQ API, or whether all publications are delivered to this subscription.

The filter value is one of the following:

***YES** Publications are only delivered to this subscription in response to an MQSUBRQ API.

***NO** All publications on the topic are delivered to this subscription.

***SELECTOR**
The SQL 92 selector string to be applied to messages published on the named topic to select whether they are eligible for this subscription.

The filter value is the selector string.

***SUBSCOPE**
Whether this subscription should be forwarded to other brokers.

The filter value is one of the following:

***ALL** The subscription will be forwarded to all queue managers directly connected via a publish / subscribe collective or hierarchy.

***QMGR**
The subscription will only forward messages published on the topic within this queue manager.

***SUBID**
The subsciption identifier associated with the subscription.

The filter value is the 48 character hexadecimal string representing the 24 byte subscription identifier.

***TOPICOBJ**
The topic object associated with the subscription.

The filter value is the name of the topic object.

***TOPICSTR**
The topic string associated with the subscription.

The filter value is the topic string.

***USERDATA**
The user data associated with the subscription.

The filter value is the user data.

***VARUSER**
Whether user profiles other than the creator of the subscription can connect to it.

The filter value is one of the following:

***ANY** Any user profiles can connect to the subsciption.

***FIXED**
Only the user profile that created the subscription can connect to it.

***WSCHEMA**
The schema to be used when interpreting wild card characters in the topic string.

The filter value is one of the following:

***TOPIC**
Wildcard characters represent portions of the topic hierarchy.

***CHAR**
Wildcard characters represent portions of strings.

Examples

None

Error messages

Unknown

Work with MQ Service object (WRKMQMSVC)

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

The Work with MQ Service objects (WRKMQMSVC) command allows you to work with multiple service objects that are defined on the local queue manager.

This enables you to start, stop, change, copy, create, delete, display, and display and change authority to an MQ service object.

Parameters

Keyword	Description	Choices	Notes
SVCNAME	Service name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *ALTTIME, *CONTROL, *ENDARG, *ENDCMD, *STDERR, *STDOUT, *STRARG, *STRCMD, *TEXT, *TYPE	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Service name (SVCNAME)

The name or names of the service objects.

The possible values are:

***ALL or ***
All service objects are selected.

generic-service-name
The generic name of the service objects. A generic name is a character string followed by an asterisk (*). For example ABC*, it selects all service objects having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

service-name
Specify the name of a single service object.

Message Queue Manager name (MQMNAME)

Specifies the name of the queue manager.

The possible values are:

***DFT** Use the default queue manager.

queue-manager-name
The name of a message queue manager.

Filter command (WHERE)

This parameter can be used to selectively display only those service objects with particular service attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***ALTDATE**

The date on which the definition or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

***ALTTIME**
The time at which the definition or information was last altered.

The filter value is the time in the form hh:mm:ss.

***CONTROL**
Whether the service is started and stopped with the queue manager.

The filter value is one of the following:

***MANUAL**
The service is not automatically started or stopped.

***QMGR**
The service is started and stopped as the queue manager is started and stopped.

***STARTONLY**
The service is started as the queue manager is started, is not be requested to stop when the queue manager is stopped.

***ENDARG**
The arguments passed to the end program when the service is requested to stop.

The filter value is the arguments string.

***ENDCMD**
The name of the executable to run when the service is requested to stop.

The filter value is the program name string.

***STDERR**
The standard error path.

The filter value is the path name.

***STDOUT**
The standard output path.

The filter value is the path name.

***STRARG**
The arguments passed to the program at startup.

The filter value is the arguments string.

***STRCMD**
The name of the program to run.

The filter value is the program name string.

***TEXT**
Descriptive comment.

The filter value is the text description of the service.

***TYPE** Mode in which to run service.

The filter value is one of the following:

***CMD** When started the command is executed but no status is collected or displayed.

***SVR** The status of the executable started is monitored and displayed.

Examples

None

Error messages

Unknown

Work with MQ Topics (WRKMQMTOP)

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

The Work with MQ Topics (WRKMQMTOP) command allows you to work with multiple topic objects that are defined on the local queue manager. This enables you to copy, change, display, delete, display authority, edit authority, record and recover an MQ topic object.

Parameters

Keyword	Description	Choices	Notes
TOPNAME	Topic name	*Character value*, ***ALL**	Optional, Positional 1
MQMNAME	Message Queue Manager name	*Character value*, ***DFT**	Optional, Positional 2

Keyword	Description	Choices	Notes
WHERE	Filter command	Single values: ***NONE** Other values: *Element list*	Optional, Positional 3
	Element 1: Filter keyword	*ALTDATE, *ALTTIME, *DFTMSGPST, *DFTPTY, *DFTPUTRESP, *DURSUB, *MGDDURMDL, *MGDNDURMDL, *NPMSGDLV, *PMSGDLV, *PUBENBL, *SUBENBL, *TEXT, *TOPNAME, *TOPICSTR, *WILDCARD	
	Element 2: Filter operator	*GT, *LT, *EQ, *NE, *GE, *LE, *LK, *NL, *CT, *EX, *CTG, *EXG	
	Element 3: Filter value	*Character value*	

Topic name (TOPNAME)

Specifies the name or names of the topic objects.

The possible values are:

***ALL** All topic objects are selected.

generic-topic-name
Specify the generic name of the MQ topic objects. A generic name is a character string followed by an asterisk (*). For example ABC*, it selects all topic objects having names that start with the character string.

You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.

You cannot select all the upper case and lower case versions of a generic name on a single panel, without requesting all the names.

topic-name
Specify the name of the MQ topic object.

Message Queue Manager name (MQMNAME)

Specifies the name of the Queue Manager.

The possible values are:

***DFT** Use the default Queue Manager.

queue-manager-name
The name of a Queue Manager.

Filter command (WHERE)

This parameter can be used to selectively display only those topices with particular topic attributes.

The parameter takes three arguments, a keyword, an operator and a value.

Generic strings are allowed for values which are names.

The operator can take one of the following values:

***GT** Greater than.

Applicable to integer and non-generic string values.

***LT** Less than.

Applicable to integer and non-generic string values

***EQ** Equal to.

Applicable to integer and non-generic string values.

***NE** Not equal to.

Applicable to integer and non-generic string values.

***GE** Greater than or equal to.

Applicable to integer and non-generic string values.

***LE** Less than or equal to.

Applicable to integer and non-generic string values.

***LK** Like.

Applicable to generic string values.

***NL** Not like.

Applicable to generic string values.

***CT** Contains.

Applicable to non-generic list values.

***EX** Excludes.

Applicable to non-generic list values.

***CTG** Contains generic.

Applicable to generic list values.

***EXG** Excludes generic.

Applicable to generic list values.

The keyword can take one of the following values:

***ALTDATE**
The date on which the object or information was last altered.

The filter value is the date in the form yyyy-mm-dd.

***ALTTIME**
The time at which the object or information was last altered.

The filter value is the time in the form hh:mm:ss.

***DFTMSGPST**
The default persistence for messages associated with this topic.

The filter value is one of the following:

***ASPARENT**
Default persistence for messages is inherited from the parent topic.

***NO** Messages associated with this topic are lost across a restart of the queue manager.

***YES** Messages associated with this topic survive a restart of the queue manager.

***DFTPUTRESP**
Default Put Response.

The filter value is one of the following:

***ASPARENT**
The default response type is based on the setting of the first parent administrative node found in the topic tree relating to this topic.

***SYNC**
Put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are issued as if MQPMO_SYNC_RESPONSE had been specified instead.

***ASYNC**
Put operations to the queue that specify MQPMO_RESPONSE_AS_Q_DEF are always issued as if MQPMO_ASYNC_RESPONSE had been specified instead.

***DFTPTY**
Default priority for messages associated with this topic.

The filter value is the integer priority value.

***DURSUB**
Specifies whether, or not, the topic permits durable subscriptions.

The filter value is one of the following:

***ASPARENT**
This topic behaves in the same way as the parent topic.

***NO** This topic does not permit durable subscriptions.

***YES** This topic does permit durable subscriptions.

***MGDDURMDL**
The name of the model queue for managed durable subscriptions.

The filter value is the name of the queue.

***MGDNDURMDL**
The name of the model queue for managed non-durable subscriptions.

The filter value is the name of the queue.

***NPMSGDLV**
Specifies the delivery mechanism for non-persistent messages published to this topic.

The filter value is one of the following:

***ALL** All non-persistent messages are published to this topic.

***ALLDUR**
All durable non-persistent messages are published to this topic.

***ALLAVAIL**
All available non-persistent messages are published to this topic.

***ASPARENT**
This topic behaves in the same way as the parent topic.

***PMSGDLV**
Specifies the delivery mechanism for persistent messages published to this topic.

The filter value is one of the following:

***ALL** All persistent messages are published to this topic.

***ALLDUR**
All durable persistent messages are published to this topic.

***ALLAVAIL**
All available persistent messages are published to this topic.

***ASPARENT**
This topic behaves in the same way as the parent topic.

***PUBENBL**
Specifies whether, or not, the topic allows publications.

The filter value is one of the following:

***ASPARENT**
This topic behaves in the same way as the parent topic.

***NO** This topic does not have publication enabled.

***YES** This topic does have publication enabled.

***SUBENBL**
Specifies whether, or not, the topic allows subscriptions.

The filter value is one of the following:

***ASPARENT**
This topic behaves in the same way as the parent topic.

***NO** This topic does not allow subscriptions.

***YES** This topic allows subscriptions.

***TEXT**
Descriptive comment.

The filter value is the text description of the topic.

***TOPNAME**
The name of the topic.

The filter value is the name of the topic.

***TOPICSTR**
The topic string, used to identify the topic node.

The filter value is a character string.

***WILDCARD**
Specifies the behaviour of wildcard subscriptions with respect to this topic.

The filter value is one of the following:

***PASSTHRU**
Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will receive publications made to this topic and to topic strings more specific than this topic.

***BLOCK**
Subscriptions made to a wildcarded topic less specific than the topic string at this topic object will not receive publications made to this topic or to topic strings more specific than this topic.

Examples

None

Error messages

Unknown

Work with MQ Transactions (WRKMQMTRN)

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

The work with MQ transactions (WRKMQMTRN) command lists details of internally or externally coordinated in-doubt transactions.

Parameters

Keyword	Description	Choices	Notes
TYPE	Transaction type	***ALL** , *EXT, *INT, *MQI, *XA, *OS400	Optional, Positional 1

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

Transaction type (TYPE)

Specifies the type of in-doubt transactions.

***ALL** Requests details of all the in-doubt transactions.

***EXT** Requests details of externally coordinated, in-doubt transactions. Such transactions are those for which WebSphere MQ has been asked to prepare to commit, but has not yet been informed of the transaction outcome.

***INT** Requests details of internally coordinated, in-doubt transactions. Such transactions are those for which each resource manager has been asked to prepare to commit, but WebSphere MQ has yet to inform the resource managers of the transaction outcome.

Message Queue Manager name (MQMNAME)

Specifies the name of the message queue manager.

The possible values are:

***DFT** Use the default queue manager.

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

Examples

None

Error messages

Unknown

Chapter 12. WebSphere MQ names and default objects

This appendix describes the requirements for WebSphere MQ object names, queue manager name transformations, and lists the system default objects.

WebSphere MQ object names

The names of the following WebSphere MQ objects can have up to 48 single-byte characters:

- Queue managers
- Queues
- Process definitions
- Namelists

The names of channels are restricted to 20 single-byte characters.

The characters that can be used for all WebSphere MQ names are:

- Uppercase A–Z
- Numerics 0–9
- Period (.)
- Underscore (_)
- Lowercase a–z (see note 1)
- Forward slash (/) (see note 1)
- Percent sign (%) (see note 1)

Note:

1. 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.)

   You cannot use lowercase characters on systems using EBCDIC Katakana.
2. Leading or embedded blanks are not allowed.

Understanding WebSphere MQ queue manager library names

When a queue manager is created, WebSphere MQ associates a queue manager library with it. This queue manager library is given a unique name, no more than 10 characters long, largely based on the user defined queue manager name. Both the queue manager, and the queue manager library are placed in to a directory that is also based on the queue manager name with the prefix `/QIBM/UserData/mqm`. An example of a queue manager, queue manager library, and directory follows:

Table 14. Queue manager library names

Queue manager name	ORANGE
Queue manager library name	QMORANGE
Directory	`/QIBM/UserData/mqm/ORANGE`

All queue manager names and queue manager library names are written to stanzas in the file /QIBM/UserData/mqm/mqs.ini.

Understanding WebSphere MQ IFS directories and files

The i5/OS Integrated File System (IFS) is used extensively by WebSphere MQ to store data. For more information about the IFS see the *Integrated File System Introduction*.

Each WebSphere MQ object (queue, queue manager and so on) is represented by a file. Because object names are not necessarily valid file names, the queue manager converts the object name into a valid file name where necessary.

The path to a queue manager directory is formed from the following:

- A prefix, which is defined in the queue manager configuration file, qm.ini. The default prefix is /QIBM/UserData/mqm.
- A literal, qmgrs.
- A coded queue manager name, which is the queue manager name transformed into a valid directory name. For example, the queue manager queue/manager is represented by queue&manager.

This process is referred to as name transformation.

IFS queue manager name transformation

In WebSphere MQ, you can give a queue manager a name containing up to 48 characters.

For example, you can name a queue manager QUEUE/MANAGER/ACCOUNTING/SERVICES. In the same way that a library is created for each queue manager, each queue manager is also represented by a file. Because of variant codepoints in EBCDIC, there are limitations to the characters that can be used in the name. As a result, the names of IFS files representing objects are automatically transformed to meet the requirements of the file system.

Using the example of a queue manager with the name queue/manager, transforming the character / to &, and assuming the default prefix, the queue manager name in WebSphere MQ for i5/OS becomes /QIBM/UserData/mqm/qmgrs/queue&manager.

Object name transformation

Object names are not necessarily valid file system names, so the object names might need to be transformed. The method used is different from that for queue manager names because, although there only a few queue manager names for each machine, there can be a large number of other objects for each queue manager. Only process definitions, queues, and namelists are represented in the file system; channels are not affected by these considerations.

When a new name is generated by the transformation process, there is no simple relationship with the original object name. You can use the DSPMQMOBJN command to view the transformed names for WebSphere MQ objects.

System and default objects

When you create a queue manager using the **CRTMQM** command, the system objects and the default objects are created automatically.

- The system objects are those WebSphere MQ objects required for the operation of a queue manager or channel.
- The default objects define all the attributes of an object. When you create an object, such as a local queue, any attributes that you do not specify explicitly are inherited from the default object.

The following tables list the system and default objects created by **CRTMQM**:

- Table 15 lists the system and default queue objects.
- Table 16 on page 684 lists the system and default channel objects.
- Table 17 on page 684 gives the system and default authentication information object.
- Table 18 on page 684 gives the system and default listener object.
- Table 19 on page 685 gives the system and default namelist object.
- Table 20 on page 685 gives the system and default process object.
- Table 21 on page 685 gives the system and default service object.

Table 15. System and default objects: queues

Object name	Description
SYSTEM.ADMIN.CHANNEL.EVENT	Event queue for channels.
SYSTEM.ADMIN.COMMAND.QUEUE	Administration command queue. Used for remote MQSC commands and PCF commands.
SYSTEM.ADMIN.PERFM.EVENT	Event queue for performance events.
SYSTEM.ADMIN.QMGR.EVENT	Event queue for queue manager events.
SYSTEM.AUTH.DATA.QUEUE	Used by the object authority manager (OAM).
SYSTEM.CHANNEL.INITQ	Channel initiation queue.
SYSTEM.CHANNEL.SYNCQ	The queue that holds the synchronization data for channels.
SYSTEM.CICS.INITIATION.QUEUE	Default CICS initiation queue.
SYSTEM.CLUSTER.COMMAND.QUEUE	The queue used to carry messages to the repository queue manager.
SYSTEM.CLUSTER.REPOSITORY.QUEUE	The queue used to store all repository information.
SYSTEM.CLUSTER.TRANSMIT.QUEUE	The transmission queue for all messages to all clusters.
SYSTEM.DEAD.LETTER.QUEUE	Dead-letter (undelivered message) queue.
SYSTEM.DEFAULT.ALIAS.QUEUE	Default alias queue.
SYSTEM.DEFAULT.AUTHINFO.CRLLDAP	Default authentication information definition.
SYSTEM.DEFAULT.INITIATION.QUEUE	Default initiation queue.
SYSTEM.DEFAULT.LOCAL.QUEUE	Default local queue.
SYSTEM.DEFAULT.MODEL.QUEUE	Default model queue.

Table 15. System and default objects: queues (continued)

Object name	Description
SYSTEM.DEFAULT.REMOTE.QUEUE	Default remote queue.
SYSTEM.MQEXPLORER.REPLY.MODEL	WebSphere MQ Explorer reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to the WebSphere MQ Explorer.
SYSTEM.MQSC.REPLY.QUEUE	MQSC command reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to remote MQSC commands.
SYSTEM.PENDING.DATA.QUEUE	Support deferred messages in JMS.

Table 16. System and default objects: channels

Object name	Description
SYSTEM.AUTO.RECEIVER	Dynamic receiver channel.
SYSTEM.AUTO.SVRCONN	Dynamic server-connection channel.
SYSTEM.DEF.CLNTCONN	Default client connection channel, used to supply default values for any attributes not specified when a CLNTCONN channel is created on a queue manager.
SYSTEM.DEF.CLUSRCVR	Default receiver channel for the cluster used to supply default values for any attributes not specified when a CLUSRCVR channel is created on a queue manager in the cluster.
SYSTEM.DEF.CLUSSDR	Default sender channel for the cluster used to supply default values for any attributes not specified when a CLUSSDR channel is created on a queue manager in the cluster.
SYSTEM.DEF.RECEIVER	Default receiver channel.
SYSTEM.DEF.REQUESTER	Default requester channel.
SYSTEM.DEF.SENDER	Default sender channel.
SYSTEM.DEF.SERVER	Default server channel.
SYSTEM.DEF.SVRCONN	Default server-connection channel.

Table 17. System and default objects: authentication information objects

Object name	Description
SYSTEM.DEFAULT.AUTHINFO. CRLLDAP	Default authentication information object.

Table 18. System and default objects: listeners

Object name	Description
SYSTEM.DEFAULT.LISTENER.TCP	Default listener for TCP transport.
SYSTEM.DEFAULT.LISTENER.LU62	Default listener for LU62 transport.
SYSTEM.DEFAULT.LISTENER.NETBIOS	Default listener for NETBIOS transport.
SYSTEM.DEFAULT.LISTENER.SPX	Default listener for SPX transport.

Table 19. System and default objects: namelists

Object name	Description
SYSTEM.DEFAULT.NAMELIST	Default namelist definition.

Table 20. System and default objects: processes

Object name	Description
SYSTEM.DEFAULT.PROCESS	Default process definition.

Table 21. System and default objects: services

Object name	Description
SYSTEM.DEFAULT.SERVICE	Default service.
SYSTEM.BROKER	Publish/Subscribe broker.

Chapter 13. Sample resource definitions

This appendix contains the AMQSAMP4 sample i5/OS CL program.

```
/*********************************************************************/
 /*                                                                   */
 /* Program name: AMQSAMP4                                            */
 /*                                                                   */
 /* Description: Sample CL program defining MQM queues                */
 /*              to use with the sample programs                      */
 /*              Can be run, with changes as needed, after            */
 /*              starting the MQM                                     */
 /*                                                                   */
 /* <N_OCO_COPYRIGHT>                                                 */
 /* Licensed Materials - Property of IBM                              */
 /*                                                                   */
 /* 63H9336                                                           */
 /* (c) Copyright IBM Corp. 1993, 2005 All Rights Reserved.           */
 /*                                                                   */
 /* US Government Users Restricted Rights - Use, duplication or       */
 /* disclosure restricted by GSA ADP Schedule Contract with           */
 /* IBM Corp.                                                         */
 /* <NOC_COPYRIGHT>                                                   */
 /*                                                                   */
 /*********************************************************************/
 /*                                                                   */
 /* Function:                                                         */
 /*                                                                   */
 /*                                                                   */
 /*   AMQSAMP4 is a sample CL program to create or reset the          */
 /*   MQI resources to use with the sample programs.                  */
 /*                                                                   */
 /*   This program, or a similar one, can be run when the MQM         */
 /*   is started - it creates the objects if missing, or resets       */
 /*   their attributes to the prescribed values.                      */
 /*                                                                   */
 /*                                                                   */
 /*                                                                   */
 /*                                                                   */
 /*   Exceptions signaled:  none                                      */
 /*   Exceptions monitored: none                                      */
 /*                                                                   */
 /*   AMQSAMP4 takes a single parameter, the Queue Manager name       */
 /*                                                                   */
 /*********************************************************************/
            QSYS/PGM PARM(&QMGRNAME)

 /*********************************************************************/
 /*   Queue Manager Name Parameter                                    */
 /*********************************************************************/
            QSYS/DCL VAR(&QMGRNAME) TYPE(*CHAR)

 /*********************************************************************/
 /*       EXAMPLES OF DIFFERENT QUEUE TYPES                           */
 /*                                                                   */
 /*   Create local, alias and remote queues                           */
 /*                                                                   */
 /*   Uses system defaults for most attributes                        */
 /*                                                                   */
 /*********************************************************************/
 /*   Create a local queue                                            */
            CRTMQMQ     QNAME('SYSTEM.SAMPLE.LOCAL')                 +
                        MQMNAME(&QMGRNAME)                           +
                        QTYPE(*LCL)  REPLACE(*YES)                   +
```

```
                                                                      +
                          TEXT('Sample local queue') /* description */+
                          SHARE(*YES)                /* Shareable   */+
                          DFTMSGPST(*YES)  /* Persistent messages OK  */

/*   Create an alias queue                                            */
            CRTMQMQ       QNAME('SYSTEM.SAMPLE.ALIAS')                +
                          MQMNAME(&QMGRNAME)                          +
                          QTYPE(*ALS)  REPLACE(*YES)                  +
                                                                      +
                          TEXT('Sample alias queue')                  +
                          DFTMSGPST(*YES) /* Persistent messages OK */+
                          TGTQNAME('SYSTEM.SAMPLE.LOCAL')

/*   Create a remote queue - in this case, an indirect reference      */
/*      is made to the sample local queue on OTHER queue manager      */
            CRTMQMQ       QNAME('SYSTEM.SAMPLE.REMOTE')               +
                          MQMNAME(&QMGRNAME)                          +
                          QTYPE(*RMT)  REPLACE(*YES)                  +
                                                                      +
                          TEXT('Sample remote queue')/* description */+
                          DFTMSGPST(*YES) /* Persistent messages OK */+
                          RMTQNAME('SYSTEM.SAMPLE.LOCAL')             +
                          RMTMQMNAME(OTHER)   /* Queue is on OTHER    */

/*   Create a transmission queue for messages to queues at OTHER      */
/*   By default, use remote node name                                 */
            CRTMQMQ       QNAME('OTHER') /* transmission queue name */+
                          MQMNAME(&QMGRNAME)                          +
                          QTYPE(*LCL)  REPLACE(*YES)  +
                          TEXT('Transmision queue to OTHER')  +
                          USAGE(*TMQ)    /* transmission queue      */

/********************************************************************/
/*       SPECIFIC QUEUES AND PROCESS USED BY SAMPLE PROGRAMS        */
/*                                                                  */
/*   Create local queues used by sample programs                    */
/*   Create MQI process associated with sample initiation queue     */
/*                                                                  */
/********************************************************************/
/*   General reply queue                                            */
            CRTMQMQ       QNAME('SYSTEM.SAMPLE.REPLY')                +
                          MQMNAME(&QMGRNAME)                          +
                          QTYPE(*LCL)  REPLACE(*YES)                  +
                                                                      +
                          TEXT('General reply queue')                 +
                          DFTMSGPST(*NO) /* Not Persistent         */

/*   Queue used by AMQSINQ4                                          */
            CRTMQMQ       QNAME('SYSTEM.SAMPLE.INQ')                  +
                          MQMNAME(&QMGRNAME)                          +
                          QTYPE(*LCL)  REPLACE(*YES)                  +
                                                                      +
                          TEXT('Queue for AMQSINQ4')                  +
                          SHARE(*YES)                /* Shareable   */+
                          DFTMSGPST(*NO) /* Not Persistent          */+
                                                                       +
                          TRGENBL(*YES)  /* Trigger control on      */+
                          TRGTYPE(*FIRST)/* Trigger on first message*/+
                          PRCNAME('SYSTEM.SAMPLE.INQPROCESS')         +
                          INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*   Queue used by AMQSSET4                                          */
            CRTMQMQ       QNAME('SYSTEM.SAMPLE.SET')                  +
                          MQMNAME(&QMGRNAME)                          +
                          QTYPE(*LCL)  REPLACE(*YES)                  +
                                                                      +
```

```
                                TEXT('Queue for AMQSSET4')                +
                                SHARE(*YES)               /* Shareable  */ +
                                DFTMSGPST(*NO)/* Not Persistent          */ +
                                                                          +
                                TRGENBL(*YES)  /* Trigger control on     */ +
                                TRGTYPE(*FIRST)/* Trigger on first message*/+
                                PRCNAME('SYSTEM.SAMPLE.SETPROCESS')       +
                                INITQNAME('SYSTEM.SAMPLE.TRIGGER')

 /*   Queue used by AMQSECH4                                               */
             CRTMQMQ     QNAME('SYSTEM.SAMPLE.ECHO')                      +
                         MQMNAME(&QMGRNAME)                               +
                         QTYPE(*LCL)  REPLACE(*YES)                       +
                                                                          +
                         TEXT('Queue for AMQSECH4')                       +
                         SHARE(*YES)               /* Shareable  */ +
                         DFTMSGPST(*NO)/* Not Persistent          */ +
                                                                          +
                         TRGENBL(*YES)  /* Trigger control on     */ +
                         TRGTYPE(*FIRST)/* Trigger on first message*/+
                         PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS')             +
                         INITQNAME('SYSTEM.SAMPLE.TRIGGER')

 /*   Initiation Queue used by AMQSTRG4, sample trigger process     */
             CRTMQMQ     QNAME('SYSTEM.SAMPLE.TRIGGER') +
                         MQMNAME(&QMGRNAME)                               +
                         QTYPE(*LCL)  REPLACE(*YES)   +
                         TEXT('Trigger queue for sample programs')

 /*   MQI Processes associated with triggered sample programs       */
 /*                                                                 */
 /*****    Note - there are versions of the triggered samples  ******/
 /*****      in different languages - set APPID for these      ******/
 /*****      process to the variation you want to trigger      ******/
 /*                                                                 */
             CRTMQMPRC   PRCNAME('SYSTEM.SAMPLE.INQPROCESS')              +
                         MQMNAME(&QMGRNAME)                               +
                         REPLACE(*YES)                                    +
                                                                          +
                         TEXT('Trigger process for AMQSINQ4')             +
                         ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here            **/                      +
                         APPID('QMQM/AMQSINQ4')    /* C               +
                /*       APPID('QMQM/AMQ0INQ4')    /* COBOL */       +
                /*       APPID('QMQM/AMQ3INQ4')    /* RPG - ILE */

             CRTMQMPRC   PRCNAME('SYSTEM.SAMPLE.SETPROCESS')              +
                         MQMNAME(&QMGRNAME)                               +
                         REPLACE(*YES)                                    +
                                                                          +
                         TEXT('Trigger process for AMQSSET4')             +
                         ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here            **/                      +
                         APPID('QMQM/AMQSSET4')    /* C     */       +
                /*       APPID('QMQM/AMQ0SET4')    /* COBOL */       +
                /*       APPID('QMQM/AMQ3SET4')    /* RPG - ILE */

             CRTMQMPRC   PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS')             +
                         MQMNAME(&QMGRNAME)                               +
                         REPLACE(*YES)                                    +
                                                                          +
                         TEXT('Trigger process for AMQSECH4')             +
                         ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here            **/                      +
                         APPID('QMQM/AMQSECH4')    /* C     */       +
                /*       APPID('QMQM/AMQ0ECH4')    /* COBOL */       +
                /*       APPID('QMQM/AMQ3ECH4')    /* RPG - ILE */
```

```
|
|                     /*********************************************************************/
|                     /*                                                                   */
|                     /*   Normal return.                                                  */
|                     /*                                                                   */
|                     /*********************************************************************/
|                                SNDPGMMSG   MSG('AMQSAMP4 Completed creating sample +
|                                            objects for ' *CAT &QMGRNAME)
|                                RETURN
|                                ENDPGM
|
|                     /*********************************************************************/
|                     /*                                                                   */
|                     /* END OF AMQSAMP4                                                   */
|                     /*                                                                   */
|                     /*********************************************************************/
```

Chapter 14. Quiescing WebSphere MQ for i5/OS

This section explains how to quiesce (end gracefully) WebSphere MQ for i5/OS

Quiescing WebSphere MQ for i5/OS

Instructions

To quiesce WebSphere MQ for i5/OS:

1. Sign on to a new interactive WebSphere MQ for i5/OS session, ensuring that you are not accessing any objects.
2. Ensure that you have:
   - *ALLOBJ authority , or object management authority for the QMQM library
   - Sufficient authority to use the ENDSBS command
3. Advise all users that you are going to stop MQSeries for AS/400®.
4. How you then proceed depends on whether you want to shut down (quiesce) a single queue manager (where others might exist) (see "Shutting down a single queue manager") or all the queue managers (see "Shutting down all queue managers" on page 693).

ENDMQM parameter ENDCCTJOB(*YES)

The ENDMQM parameter ENDCCTJOB(*YES) works differently in WebSphere MQ for i5/OS V6.0 and later compared to previous versions

On previous versions, when you specify ENDCCTJOB(*YES), MQ forcibly terminates your applications for you.

On WebSphere MQ for i5/OS V6.0 or later, when you specify ENDCCTJOB(*YES), your applications are not terminated but are instead disconnected from the queue manager.

If you specify ENDCCTJOB(*YES) and you have applications that are not written to detect that a queue manager is ending, the next time a new MQI call is issued, the call will return with a MQRC_CONNECTION_BROKEN (2009) error.

As an alternative to using ENDCCTJOB(*YES), use the parameter ENDCCTJOB(*NO) and use WRKMQM option 22 (Work with jobs) to manually end any application jobs that will prevent a queue manager restart.

Shutting down a single queue manager

There are three types of shutdown:
- "Planned shutdown" on page 692
- "Unplanned shutdown" on page 692
- "Shutdown under abnormal conditions" on page 692

In the procedures that follow, we use a sample queue manager name of `QMgr1` and a sample subsystem name of `SUBX`. Replace these with your own.

Planned shutdown

Planned shutdown of a queue manager on i5/OS

1. Prior to shutdown, execute:

   ```
   RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
   ```

2. To shut down the queue manager, execute:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
   ```

   If QMgr1 does not end, the channel or applications are probably busy.

3. If you need to shut down QMgr1 immediately, execute the following:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
   ENDCCTJOB(*YES) TIMEOUT(15)
   ```

Unplanned shutdown

1. To shut down the queue manager, execute:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
   ```

   If QMgr1 does not end, the channel or applications are probably busy.

2. If you need to shut down QMgr1 immediately, execute the following:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
   ENDCCTJOB(*YES) TIMEOUT(15)
   ```

Shutdown under abnormal conditions

1. To shut down the queue manager, execute:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
   ```

   If QMgr1 does not end, continue with step 3 providing that:

   - QMgr1 is in its own subsystem, or
   - You can end all queue managers that share the same subsystem as QMgr1. Use the unplanned shutdown procedure for all such queue managers.

2. When you have taken all the steps in the procedure for all the queue managers sharing the subsystem (`SUBX` in our examples), execute:

   ```
   ENDSBS SUBX *IMMED
   ```

   If this command fails to complete, shut down all queue managers, using the unplanned shutdown procedure, and IPL your machine.

   Warning: Do not use `ENDJOBABN` for MQSeries or WebSphere MQ jobs that fail to end as result of `ENDJOB` or `ENDSBS`, unless you are prepared to IPL your machine immediately after.

3. Start the subsystem by executing:

   ```
   STRSBS SUBX
   ```

4. Shut the queue manager down immediately, by executing:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
   ENDCCTJOB(*YES) TIMEOUT(10)
   ```

5. Restart the queue manager by executing:

   ```
   STRMQM MQMNAME(QMgr1)
   ```

   If this fails, and you:

   - Have restarted your machine with an IPL, or
   - Have only a single queue manager

   Tidy up MQSeries or WebSphere MQ shared memory by executing:

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

before repeating step 5.

If the queue manager restart takes more than a few seconds, WebSphere MQ will add status messages intermittently to the job log detailing the start up progress. For more information on these messages see WebSphere MQ Messages.

If you still have problems restarting your queue manager, contact IBM support. Any further action you might take could damage the queue manager, leaving MQSeries or WebSphere MQ unable to recover.

Shutting down all queue managers

There are three types of shutdown:

- "Planned shutdown"
- "Unplanned shutdown"
- "Shutdown under abnormal conditions" on page 694

The procedures are almost the same as for a single queue manager, but using *ALL instead of the queue manager name where possible, and otherwise using a command repeatedly using each queue manager name in turn. Throughout the procedures, we use a sample queue manager name of QMgr1 and a sample subsystem name of SUBX. Replace these with your own.

Planned shutdown

1. One hour before shutdown, execute:

   ```
   RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
   ```

   Repeat this for every queue manager that you want to shut down.
2. To shut down the queue manager, execute:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
   ```

   Repeat this for every queue manager that you want to shut down; separate commands can run in parallel.

   If any queue manager does not end within a reasonable time (for example 10 minutes), proceed to step 3.
3. To shut down all queue managers immediately, execute the following:

   ```
   ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
   ENDCCTJOB(*YES) TIMEOUT(15)
   ```

Unplanned shutdown

1. To shut down a queue manager, execute:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
   ```

   Repeat this for every queue manager that you want to shut down; separate commands can run in parallel.

   If queue managers do not end, the channel or applications are probably busy.
2. If you need to shut down the queue managers immediately, execute the following:

   ```
   ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
   ENDCCTJOB(*YES) TIMEOUT(15)
   ```

Shutdown under abnormal conditions

1. To shut down the queue managers, execute:

   ```
   ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
   ```

   Repeat this for every queue manager that you want to shut down; separate commands can run in parallel.
2. End the subsystems (SUBX in our examples), by executing:

   ```
   ENDSBS SUBX *IMMED
   ```

   Repeat this for every subsystem that you want to shut down; separate commands can run in parallel.

   If this command fails to complete, IPL your machine.

   Warning: Do not use ENDJOBABN for MQSeries or WebSphere MQ jobs that fail to end as result of ENDJOB or ENDSBS, unless you are prepared to IPL your machine immediately after.
3. Start the subsystems by executing:

   ```
   STRSBS SUBX
   ```

   Repeat this for every subsystem that you want to start.
4. Shut the queue managers down immediately, by executing:

   ```
   ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
   ENDCCTJOB(*YES) TIMEOUT(15)
   ```
5. Restart the queue managers by executing:

   ```
   STRMQM MQMNAME(QMgr1)
   ```

   Repeat this for every queue manager that you want to start.

   If any queue manager restart takes more than a few seconds WebSphere MQ will show status messages intermittently detailing the start up progress. For more information on these messages see WebSphere MQ Messages.

   If you still have problems restarting any queue manager, contact IBM support. Any further action you might take could damage the queue managers, leaving MQSeries or WebSphere MQ unable to recover.

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-k,u
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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

IBM 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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

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

AIX	AS/400	CICS
CICS/400	CICS/VSE	FFST
First Failure Support Technology	i5/OS	IBM
IMS	Lotus Notes	MQSeries
MVS	MVS/ESA	OS/2
RETAIN	SupportPac	WebSphere
z/OS		

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Unix is a trademark of The Open Group in the United States, other countries, or both.

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

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

Index

A

AccountingToken field
  MQZIC structure 195
ACTION keyword, rules table 86
action keywords, rules table 86
administration
  authorizations 63
  description of 39
  introduction to 35
  local, definition of 35
  MQAI, using 39
  PCF commands 37
  queue manager name transformation 681
  remote administration, definition of 35
  understanding WebSphere MQ file names 681
  using PCF commands 37
  WebSphere MQ script (MQSC) commands 36
alias queues
  authorizations to 79
  defining alias queues 28
  working with alias queues 27
AllQueueManagers stanza, mqs.ini 127
alternate-user authority 79
AMQA000000 work management object 45
AMQAJRN work management object 45
AMQAJRNMSG work management object 45
AMQALMPX task 44
AMQCLMAA task 44
AMQCRC6B work management object 45
AMQCRS6B task 44
AMQCRSTA task 44
AMQFCXBA task 44
AMQFQPUB task 44
AMQLAA0 work management object 45
AMQPCSEA task 44
AMQRCMLA task 44
AMQRFCD4 work management object 45
AMQRMPPA task 44
AMQRRMFA task 44
AMQXSSVN task 44
AMQZFUMA task 44
AMQZLAA0 task 44
AMQZLAS0 task 44
AMQZMGR0 task 44
AMQZMUC0 task 44
AMQZMUF0 task 44
AMQZMUR0 task 44
AMQZTRCN task 44
AMQZXMA0 task 44
AMQZXMA0 work management object 45
application design, performance considerations 122
application programming errors, examples of 106
application programs
  receiving messages 2
  retrieving messages from queues 3
  sending messages 2
  time-independent applications 1
application queues
  creating and copying, restrict access to 79
  defining application queues for triggering 31
  description of 7
ApplicationContext parameter
  authenticate user call 151
APPLIDAT keyword, rules table 85
ApplIdentityData field
  MQZIC structure 195
ApplName field
  MQZAC structure 186
APPLNAME keyword, rules table 85
APPLTYPE keyword, rules table 85
attributes
  changing local queue attributes 26
  queue manager 24
  queues 6
authority
  alternate-user 79
  context authority 80
authority data
  WRKMQMAUT command 76
  WRKMQMAUTD command 77
Authority field
  MQZAD structure 190
Authority parameter
  check authority call 155
  get authority call 169
  get explicit authority call 172
  set authority call 182
authority profiles
  working with 76
  working without 75
AuthorityBuffer parameter
  enumerate authority data call 164
AuthorityBufferLength parameter
  enumerate authority data call 164
AuthorityDataLength parameter
  enumerate authority data call 164
authorization service
  component 145
  defining 146
  specifying the installed 75
  stanza 146
  user interface 147
authorizations
  administration 63
  MQI 60
  specification tables 59

B

backup
  data 98
  introduction 93
  journals 98
  media images 96
  performance 103
  using journals 95

C

changing
  local queue attributes 26
  queue manager attributes 24
channels
  channel command security 81
  Channels stanza, qm.ini 130
  command security requirements 81
  description of 9
  escape command authorizations 63
  exits 12
  queue manager error log stanza, qm.ini 132
Channels stanza, qm.ini 130
characters allowed in object names 681
CharAttrCount parameter
  inquire authorization service call 177
CharAttrs parameter
  inquire authorization service call 177
checkpoints 97
CL commands
  creating a queue
    alias 24
    initiation 24
    model 24
    remote 22
    transmission 23
    using CRTMQMQ for local queues 20
    using WRKMQMQ for local queues 21
  creating WebSphere MQ objects 20
  grouping 15
  reference 197
  starting a local queue manager 19
clearing a local queue 27
client connection channel
  description of 9
clients and servers
  definition 11
  WebSphere MQ applications 11
clusters
  cluster transmission queues 7
  description of 9
  ExitProperties stanza attributes 128
command files 37
command queue 8
command queues
  command server status 41
  description of 8

command server
displaying status 41
remote administration 40
starting a command server 41
stopping a command server 41
commands, PCF 37
CompCode parameter
authenticate user call 152
check authority call 157
copy all authority call 159
delete authority call 162
enumerate authority data call 165
free user call 166
get authority call 169
get explicit authority call 172
initialize authorization service call 175
inquire authorization service call 178
MQZEP call 149
set authority call 183
terminate authorization service call 185
complete queue manager (data and journals), Restoring a 102
ComponentData parameter
authenticate user call 152
check authority call 156
copy all authority call 159
delete authority call 161
enumerate authority data call 164
free user call 166
get authority call 169
get explicit authority call 172
initialize authorization service call 174
inquire authorization service call 178
set authority call 182
terminate authorization service call 184
ComponentDataLength parameter
initialize authorization service call 174
components, installable services 141
configuration file
authorization service 146
configuration files
AllQueueManagers stanza, mqs.ini 127
Channels stanza, qm.ini 130
DefaultQueueManager stanza, mqs.ini 128
editing 125
example mqs.ini file 138
example qm.ini file 139
ExitProperties stanza, mqs.ini 128
Log stanza, qm.ini 130
mqs.ini, description of 126
priorities 126
queue manager configuration file, qm.ini 126
queue manager error log stanza, qm.ini 132
QueueManager stanza, mqs.ini 129
TCP stanza, qm.ini 133
configuring logs 130
context authority 80
Continuation parameter
authenticate user call 152
check authority call 156
copy all authority call 159
delete authority call 161
enumerate authority data call 164
free user call 166
get authority call 169
get explicit authority call 172
inquire authorization service call 178
set authority call 183
CorrelationPtr field
MQZED structure 192
MQZFP structure 193
CorrelationPtr parameter
authenticate user call 152
CorrelId, performance considerations 122
creating
dynamic (temporary) queue 3
model queue 3
predefined (permanent) queue 3
process definition 32
creating service components 145
creating WebSphere MQ objects 20

D

data
backup 98
restoring 102
data conversion
ConvEBCDICNewline attribute, AllQueueManagers stanza 127
EBCDIC NL character conversion to ASCII 127
data types, detailed description
elementary
MQHCONFIG 150
PMQFUNC 150
structure
MQZAC 185
MQZAD 188
MQZED 191
MQZFP 192
MQZIC 193
Daylight saving time 15
dead-letter header, MQDLH 83
dead-letter queues
defining a dead-letter queue 25
description of 8
default objects
introduction 11
list of 683
DefaultQueueManager stanza, mqs.ini 128
defining
alias queue 28
application queue for triggering 31
dead-letter queue 25
initiation queue 32
local queue 25
model queue 29
WebSphere MQ queues 6
deleting a local queue 27
DESTQ keyword, rules table 85
DESTQM keyword, rules table 86
diagnostic information, obtaining 113
directories, queue manager 79
display
default object attributes 26
process definitions 33
status of command server 41
distributed queuing example 33
DLQ handler
invoking 83
rules table 84
DSPMQMAUT command 75
dynamic binding 144
dynamic queues
authorizations 79
description of 3

E

EBCDIC NL character conversion to ASCII 127
EffectiveUserID field
MQZAC structure 187
ENDCCTJOB(*YES) 691
ENDMQM
ENDCCTJOB(*YES) 691
EntityDataPtr field
MQZAD structure 190
EntityDomainPtr field
MQZED structure 192
EntityName parameter
check authority call 153
get authority call 168
get explicit authority call 171
set authority call 181
EntityNamePtr field
MQZED structure 192
EntityType field
MQZAD structure 190
EntityType parameter
check authority call 154
get authority call 168
get explicit authority call 171
set authority call 181
EntryPoint parameter
MQZEP call 149
environment variables
MQSPREFIX 127
error logs
errors occurring before log established 118
example, WebSphere MQ 119
log files 117
escape PCFs 38
event queues, description of 8
examples
creating a transmission queue 23
creating an alias queue 24
creating local queues
using the CRTMQMQ command 20
using the WRKMQMQ command 21
creating remote queues
as a queue manager alias 22
as a remote queue definition 22
as an alias to a reply-to queue 23
error log, WebSphere MQ 119

examples *(continued)*
mqs.ini file 138
qm.ini file 139
ExitProperties stanza, mqs.ini 128
extending queue manager facilities 12

F

failover
performance 103
FEEDBACK keyword, rules table 86
FFST (First Failure Support Technology) 120
file names 681
files
IFS directories 682
log files, in problem determination 117
queue manager configuration 126
understanding names 681
WebSphere MQ configuration 126
Filter parameter
enumerate authority data call 163
FORMAT keyword, rules table 86
formatting trace 117
FreeParms parameter
free user call 166
function
MQZ_REFRESH_CACHE 179
Function parameter
MQZEP call 149
FWDQ keyword, rules table 87
FWDQM keyword, rules table 87

G

generic profiles 74
GRTMQMAUT command 75

H

Hconfig parameter
initialize authorization service call 174
MQZEP call 149
terminate authorization service call 184
HEADER keyword, rules table 87

I

i5/OS message queue 114
IdentityContext parameter
authenticate user call 152
initialization 143
initiation queues
defining 32
description of 7
INPUTQ keyword, rules table 84
INPUTQM keyword, rules table 84
installable service
authorization service 145
component
authenticate user 150
check authority 153
installable service *(continued)*
component *(continued)*
copy all authority 157
delete authority 160
enumerate authority data 162
free user 165
get authority 167
get explicit authority 170
initialize authorization service 173
inquire authorization service 175
MQZEP 149
set authority 181
terminate authorization service 184
Component data 143
component entry-points 142
components 142
configuring services 144
functions 142
initialization 143
interface to 148
return information 143
installable services 179
installed authorization service
specifying 75
installed authorization service, Specifying the 75
IntAttrCount parameter
inquire authorization service call 176
IntAttrs parameter
inquire authorization service call 177

J

journals
backup 98
introduction 93
managing 99
restoring 102
using 95

L

length of object names 681
libraries, using SAVLIB to save WebSphere MQ 104
listener objects
description of 10
local administration, definition of 35
local queues 24
changing queue attributes, commands to use 26
clearing 27
copying a local queue definition 26
defining 25
defining application queues for triggering 31
deleting 27
description of 5
specific queues used by WebSphere MQ 7
working with local queues 24
Log stanza, qm.ini 130
logical unit of work, definition of 13
logs
configuring 130
errors occurring before error log established 118
log files, in problem determination 117
Log stanza, qm.ini 130

M

managing objects for triggering 29
maximum line length, MQSC commands 37
media images
introduction 96
recovery 97
message length, decreasing 27
message persistence, performance considerations 122
message queuing 1
message-driven processing 1
messages
application data 2
containing unexpected information 112
definition of 2
message descriptor 2
message lengths 2
message-driven processing 1
not appearing on queues 111
operator messages 118
queuing 1
retrieval algorithms 3
retrieving messages from queues 3
sending and receiving 2
undelivered 120
model queues
creating a model queue 3
defining 29
working with 29
MQAI, description of 38
MQDLH, dead-letter header 83
MQHCONFIG 150
MQI (message-queuing interface)
authorization specification tables 59
authorizations 60
definition of 1
queue manager calls 5
receiving messages 2
sending messages 2
MQI authorizations 60
MQOPEN authorizations 60
MQOT_* values 189
MQPUT and MQPUT1, performance considerations 123
MQPUT authorizations 60
mqs.ini configuration file
AllQueueManagers stanza 127
DefaultQueueManager stanza 128
definition of 125
editing 125
ExitProperties stanza 128
priorities 126
QueueManager stanza 129
MQSC commands
authorization 63
command files, input 37

MQSC commands *(continued)*
escape PCFs 38
maximum line length 37
object attribute names 5
overview 36
MQSPREFIX, environment variable 127
MQZ_AUTHENTICATE_USER call 150
MQZ_CHECK_AUTHORITY call 153
MQZ_COPY_ALL_AUTHORITY call 157
MQZ_DELETE_AUTHORITY call 160
MQZ_ENUMERATE_AUTHORITY_DATA call 162
MQZ_FREE_USER call 165
MQZ_GET_AUTHORITY call 167
MQZ_GET_EXPLICIT_AUTHORITY call 170
MQZ_INIT_AUTHORITY call 173
MQZ_INQUIRE call 175
MQZ_REFRESH_CACHE function 179
MQZ_SET_AUTHORITY call 181
MQZ_TERM_AUTHORITY call 184
MQZAC structure 185
MQZAC_* values 186
MQZAD structure 188
MQZAD_* values 188
MQZAET_* values 190
MQZAO_* values 190
MQZAO, constants and authority 60
MQZED structure 191
MQZED_* values 191
MQZEP call 149
MQZFP structure 192
MQZFP_* values 193
MQZIC structure 193
MQZIC_* values 194
MQZSE_* values 163
MsgId, performance considerations 122
MSGTYPE keyword, rules table 86

N

namelists, description of 10
naming conventions 3
national language support
EBCDIC NL character conversion to ASCII 127
operator messages 118
NL character, EBCDIC conversion to ASCII 127

O

OAM (Object Authority Manager)
description of 52
guidelines for using 79
resources protected by 52
sensitive operations 79
object authority manager 146
object names 4
ObjectName parameter
check authority call 154
copy all authority call 158
delete authority call 161
get authority call 168
get explicit authority call 171
set authority call 182
objects
access to 51
administration of 35
attributes of 5
automation of administration tasks 37
channels, description of 9
client connection channels, description of 9
clusters, description of 9
creating 20
default object attributes, displaying 26
generic profiles 74
listeners, description of 10
local queues 5
managing objects for triggering 29
multiple queues 5
namelist, description of 10
naming conventions 4
process definitions 10
queue manager objects used by MQI calls 5
queue managers 5
queue objects, using 6
remote queues 5
service, description of 11
system default objects 11
using MQSC commands to administer 36
ObjectType field
MQZAD structure 189
ObjectType parameter
check authority call 154
copy all authority call 158
delete authority call 161
get authority call 168
get explicit authority call 171
set authority call 182
operator
commands, no response from 109
messages 118
Options parameter
initialize authorization service call 174
terminate authorization service call 184

P

pattern-matching keywords, rules table 85
PCF (programmable command format)
administration tasks 37
attributes in PCFs 38
authorization specification tables 59
automating administrative tasks using PCF 37
escape PCFs 38
MQAI, using to simplify use of 39
object attribute names 5
performance considerations
application design 122
CorrelId 122
message persistence 122
MQPUT and MQPUT1 123
MsgId 122
syncpoint 123
trace 114
variable length 123
permanent (predefined) queues 3
PERSIST keyword, rules table 86
PMQFUNC 150
predefined (permanent) queues 3
primary initialization 143
primary termination 144
problem determination
application design 122
applications 110
command errors 108
dead-letter queues 120
diagnostic information 113
distributed queues 113
error logs 117
error logs in the IFS 114
errors occurring before error log established 118
FFST (First Failure Support Technology) 120
i5/OS message queue 114
introduction 105
log files 117
message length 122
message persistence 122
messages 111
no response from commands 109
operator messages 118
performance 122
preliminary checks
network effects 108
operating system 109
problem affects all users 108
problem intermittent 108
problem occurs at specific times 109
problem that can be reproduced 107
problem characteristics 107
programming errors 106
queue problems 110
remote queues 111
sample error log 119
storage 123
syncpoint frequency 123
system history log 114
threads 123
trace 114
undelivered messages 120
unexpected message information 112
user's job log 113
WebSphere MQ job log 114
process definitions
creating 32
description of 10
displaying 33
ProcessId field
MQZAC structure 186
processing, message-driven 1
ProfileName field
MQZAD structure 189
programming errors, examples of 106

protected resources 52
PUTAUT keyword, rules table 87

Q

qm.ini configuration file
  Channels stanza 130
  definition of 126
  editing 125
  Log stanza 130
  priorities 126
  queue manager error log stanza 132
  TCP stanza 133
QMgrName parameter
  authenticate user call 151
  check authority call 153
  copy all authority call 158
  delete authority call 160
  enumerate authority data call 163
  free user call 166
  get authority call 167
  get explicit authority call 171
  initialize authorization service call 174
  inquire authorization service call 176
  set authority call 181
  terminate authorization service call 184
QMQM work management object 45
QMQMJOBD work management object 45
QMQMMSG work management object 45
QMQMRUN20 work management object 45
QMQMRUN35 work management object 45
QMQMRUN50 work management object 45
QTIMZON 15
queue manager
  ini file
    authorization service 146
  restoring complete 102
queue manager error log stanza, qm.ini 132
queue manager ini file 146
queue managers
  attributes, changing 24
  authorizations 79
  command server 40
  description of 5
  directories 79
  extending queue manager facilities 12
  name transformation 681
  object authority manager, description 52
  objects used in MQI calls 5
  qm.ini files 126
QueueManager stanza, mqs.ini 129
queues
  alias 27
  application queues 31
  attributes 6
  authorizations to 79
  changing queue attributes 26
queues *(continued)*
  clearing local queues 27
  dead-letter, defining 25
  defining WebSphere MQ queues 6
  definition of 3
  deleting a local queue 27
  dynamic (temporary) queues 3
  extending queue manager facilities 12
  initiation queues 32
  local queues 5
  local, working with 24
  model queues 3, 29
  multiple queues 5
  predefined (permanent) queues 3
  queue managers, description of 5
  queue objects, using 6
  retrieving messages from 3
  specific local queues used by WebSphere MQ 7
quiescing
  WebSphere MQ for i5/OS 691

R

REASON keyword, rules table 86
Reason parameter
  authenticate user call 152
  check authority call 157
  copy all authority call 159
  delete authority call 162
  enumerate authority data call 165
  free user call 167
  get authority call 169
  get explicit authority call 172
  initialize authorization service call 175
  inquire authorization service call 178
  MQZEP call 150
  set authority call 183
  terminate authorization service call 185
recovery
  introduction 93
  media images 96
  performance 103
  using journals 95
RefObjectName parameter
  copy all authority call 158
remote administration
  command server 40
  definition of remote administration 35
remote queues
  authorizations to 79
  examples of creating 22
  security considerations 80
reply-to queues, description of 8
REPLYQ keyword, rules table 86
REPLYQM keyword, rules table 86
Reserved field
  MQZFP structure 193
resources, updating under syncpoint control 12
restart
  introduction 93
  media images 96
restart *(continued)*
  performance 103
  using journals 95
restoring
  complete queue manager 102
  journal receivers 102
restricting access to MQM objects 51
retrieval algorithms for messages 3
RETRY keyword, rules table 88
RETRYINT keyword, rules table 84
rules table, DLQ handler
  control data entry
    INPUTQ keyword 84
    INPUTQM keyword 84
    RETRYINT keyword 84
    WAIT keyword 84
  example of 91
  patterns and actions (rules)
    ACTION keyword 86
    APPLIDAT keyword 85
    APPLNAME keyword 85
    APPLTYPE keyword 85
    DESTQ keyword 85
    DESTQM keyword 86
    FEEDBACK keyword 86
    FORMAT keyword 86
    FWDQ keyword 87
    FWDQM keyword 87
    HEADER keyword 87
    introduction 85
    MSGTYPE keyword 86
    PERSIST keyword 86
    PUTAUT keyword 87
    REASON keyword 86
    REPLYQ keyword 86
    REPLYQM keyword 86
    RETRY keyword 88
    USERID keyword 86
  processing 90
  syntax 88
RUNMQBRK task 44
RUNMQCHI task 44
RUNMQCHL task 44
RUNMQDLQ task 44
RUNMQLSR task 44
RUNMQTRM task 44
RVKMQMAUT command 75

S

saving time, Daylight 15
SAVLIB, using to save WebSphere MQ libraries 104
secondary initialization 143
secondary termination 144
secure sockets layer (SSL)
  MQSC commands 81
  protecting channels 81
security
  administration authorizations 63
  command security requirements 81
  considerations 51
  context authority 80
  MQI authorizations 60
  object authority manager (OAM) 52
  remote queues 80
  resources protected by the OAM 52

security *(continued)*
sensitive operations, OAM 79
WebSphere MQ authorities 52
security enabling interface (SEI) 145
SecurityId field
MQZED structure 192
SecurityParms parameter
authenticate user call 151
SEI (WebSphere MQ security enabling interface) 145
SelectorCount parameter
inquire authorization service call 176
SelectorReturned parameter
inquire authorization service call 177
Selectors parameter
inquire authorization service call 176
sensitive operations, OAM 79
servers 11
service component
authorization 145
creating your own 145
stanza 144
service objects
description of 11
service stanza 144
specifying the installed authorization service 75
stanza
authorization service 146
stanzas
AllQueueManagers, mqs.ini 127
Channels, qm.ini 130
DefaultQueueManager, mqs.ini 128
ExitProperties, mqs.ini 128
Log, qm.ini 130
queue manager error log, qm.ini 132
QueueManager, mqs.ini 129
TCP, qm.ini 133
StartEnumeration parameter
enumerate authority data call 163
starting a command server 41
stopping a command server 41
storage problems 123
STRMQMDLQ command 83
StrucId field
MQZAC structure 186
MQZAD structure 188
MQZED structure 191
MQZFP structure 193
MQZIC structure 194
SupportPac
MS03 15
SupportPac MS03 15
syncpoint, performance considerations 123
system default objects 11
system history log 114
system objects 683

T

tasks, WebSphere MQ 44
TCP stanza, qm.ini 133
temporary (dynamic) queues 3
termination 144
ThreadID field
MQZAC structure 186
time-independent applications 1
time, Daylight saving 15
trace data
formatting 117
lifetime of 114
selective 115
usage of
on i5/OS 114
wrapping 116
trace, performance considerations 114
transactional support, updating under syncpoint control 12
transmission queues
cluster transmission queues 7
description of 7
trigger monitor
description 30
job submission attributes 30
triggering
defining an application queue for triggering 31
description 30
managing objects for triggering 29
message-driven processing 1
setting up objects 31

U

user exits
channel exits 12
data conversion exits 12
user's job log 113
UserID field
MQZAC structure 186
USERID keyword, rules table 86
UserIdentifier field
MQZIC structure 194
using CL commands 15
using SAVLIB to save WebSphere MQ libraries 104

V

variable length, performance considerations 123
Version field
MQZAC structure 186
MQZAD structure 188
MQZED structure 192
MQZFP structure 193
MQZIC structure 194
Version parameter
initialize authorization service call 174

W

WAIT keyword, rules table 84
WebSphere MQ
creating objects 20
job log 114
security enabling interface (SEI) 145
WebSphere MQ Explorer
description of 39
required resource definitions 40
WebSphere MQ for i5/OS
backups of data 98
CL commands 15
journal management 99
journal usage 95
journals 93
media images 96
quiescing 691
recovery from media images 97
restoring a complete queue manager 102
restoring journal receivers 102
WebSphere MQ objects, creating 20
WebSphere MQ tasks 44
work management
objects 44
tasks 43
using 45
working with authority profiles 76
working without authority profiles 75
wrapping trace 116
WRKMQMAUT command 76
WRKMQMAUTD command 77

Sending your comments to IBM

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 make comments about the functions of IBM products or systems, 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, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44-1962-816151
  - From within the U.K., use 01962-816151
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

SC34-6930-01

Spine information:

WebSphere MQ for i5/OS System Administration Guide Version 7.0