MQSeries for AS/400

Administration Guide

Version 4 Release 2.1

GC33-1956-01

MQSeries for AS/400

Administration Guide

Version 4 Release 2.1

GC33-1956-01

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix E, “Notices” on page 279.

Second edition (September 1998)

This edition applies to the following product:

MQSeries for AS/400 Version 4 Release 2 Modification 1 and to any subsequent releases and modifications until otherwise indicated in new editions.

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

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

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

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

© **Copyright International Business Machines Corporation 1994, 1998. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 1994, 1998

Figures

Tables

About this book

This book defines the system administration aspects of the IBM MQSeries for AS/400 Version 4 Release 2 Modification 1 product. MQSeries for AS/400 is a member of the IBM MQSeries group of products and provides support for messaging and queuing in an IBM AS/400 V4R2 or V4R3 environment.

Who this book is for

Primarily, this book is intended for programmers or operators who manage the configuration and administration tasks for MQSeries for AS/400.

What you need to know to understand this book

To use this book, you should have a general understanding of the IBM operating system for the AS/400.

To understand this book, you do not need to have worked with message queuing products previously.

How to use this book

This book contains:

- Procedural material to help you:
  - Install and set up the product
  - Set up security
  - Work with MQSeries for AS/400 using CL commands
  - Work with MQSeries for AS/400 using the Administration utility
  - Recover from a system failure
  - Analyze any problems that arise
- Reference material for all the MQSeries for AS/400 CL commands, using railroad syntax.
- Sample resource definitions.
- Administration utility example tasks.
- Descriptions of MQSeries for AS/400 restrictions.

This book contains examples on the following topics:

- Creating various types of queue object – see “Creating MQSeries objects” on page 37.
- Using the Administration utility – see Appendix B, “Administration utility – example tasks” on page 269.

© Copyright IBM Corp. 1994, 1998

MQSeries publications

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

MQSeries cross-platform publications

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

- MQSeries for AIX V5.0
- MQSeries for AS/400 V4R2M1
- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for Digital OpenVMS V2.2
- MQSeries for HP-UX V5.0
- MQSeries for MVS/ESA V1.2
- MQSeries for OS/2 Warp V5.0
- MQSeries for SINIX and DC/OSx V2.2
- MQSeries for SunOS V2.2
- MQSeries for Sun Solaris V5.0
- MQSeries for Tandem NonStop Kernel V2.2
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1
- MQSeries for Windows NT V5.0

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

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

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

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

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

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

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

This book applies to the following MQSeries products only:

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

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

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

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

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

This book is available in softcopy only.

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

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

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

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

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

MQSeries C++ is also supported by MQSeries for AS/400 V4R2M1.

MQSeries platform-specific publications

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

MQSeries for AIX

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

MQSeries for AS/400

MQSeries for AS/400 Version 4 Release 2.1 Administration Guide, GC33-1956

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

MQSeries for AT&T GIS UNIX

MQSeries for AT&T GIS UNIX Version 2 Release 2 System Management Guide, GC33-1642

MQSeries for Digital OpenVMS

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

MQSeries for HP-UX

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

MQSeries for MVS/ESA

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

MQSeries for MVS/ESA Version 1 Release 2 Program Directory

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

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

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

MQSeries for OS/2 Warp

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

MQSeries link for R/3

| *MQSeries for R/3 Version 1.1 User’s Guide*, GC33-1934

MQSeries for SINIX and DC/OSx

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

MQSeries for SunOS

MQSeries for SunOS Version 2 Release 2 System Management Guide, GC33-1772

MQSeries for Sun Solaris

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

MQSeries for Tandem NonStop Kernel

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

MQSeries Three Tier

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

MQSeries for Windows

MQSeries for Windows Version 2.0 User’s Guide, GC33-1822

MQSeries for Windows Version 2.1 User’s Guide, GC33-1965

MQSeries for Windows NT

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

MQSeries Level 1 product publications

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

MQSeries: Concepts and Architecture, GC33-1141

MQSeries Version 1 Products for UNIX Operating Systems Messages and Codes, SC33-1754

MQSeries for SCO UNIX Version 1.4 User’s Guide, SC33-1378

MQSeries for UnixWare Version 1.4.1 User’s Guide, SC33-1379

MQSeries for VSE/ESA Version 1 Release 4 Licensed Program Specifications, GC33-1483

MQSeries for VSE/ESA Version 1 Release 4 User’s Guide, SC33-1142

Softcopy books

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

BookManager format

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

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

PostScript format

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

HTML format

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

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

The MQSeries books are also available from the MQSeries software-server home page at:

```
http://www.software.ibm.com/ts/mqseries/
```

Information Presentation Facility (IPF) format

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

Windows Help format

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

MQSeries information available on the Internet

MQSeries web site

The MQSeries product family Web site is at:

```
http://www.software.ibm.com/ts/mqseries/
```

| By following links from this Web site you can:

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

Related publications

AS/400 CL Reference Common CL Information, SC41-5722
AS/400 Backup and Recovery, SC41-5304
AS/400 Security - Reference, SC41-5302
AS/400 National Language Support, SC41-5101
AS/400 System API Reference, SC41-5801

Summary of changes

This section lists the new function in MQSeries for AS/400 V4R2M1, together with a list of the new function made available in the previous release of the product.

New function in MQSeries for AS/400 V4R2M1

MQSeries for AS/400 Version 4 Release 2 Modification 1 provides the following new function:

- Inclusion of the MQSeries dead-letter queue handler
- Inclusion of the STRMQMDLQ CL command
- Improvements to the migration procedure

Additions to the Administration Guide, GC33-1956-01

Changes to the book include:

- A new chapter , Chapter 8, “The MQSeries dead-letter queue handler” on page 87.
- Inclusion of the associated STRMQMDLQ CL command (see “STRMQMDLQ (Start MQSeries Dead-Letter Queue Handler)” on page 223).
- Minor technical and editorial improvements have been made throughout the book.

New function in MQSeries for AS/400 V4R2

MQSeries for AS/400 Version 4 Release 2 provides the following new function:

- Distribution lists, which allow a single message to be put to multiple queues using a single MQPUT or MQPUT1 call.
- Message segmentation and grouping of messages.
- Fast nonpersistent messages.
- Channel heartbeats. Heartbeat flows are passed from a sending MCA when there are no messages on the transmission queue, allowing the receiving MCA the opportunity to quiesce the channel.
- Automatic creation of channel definitions for receiver and server-connection channels.
- Static bindings for the ILE RPG programming language
- Support for MQI applications written in C++.
- Chained exits.
- Improved triggering rules.

Additions to the Administration Guide, GC33-1956-00

Changes to the book include:

- Inclusion of a section on migration from a previous release.
- Inclusion of channel heartbeats.

  Addition of HRTBTINTVL and NPMSPEED fields to the following commands:

  – CHGMQMCHL
  – CPYMQMCHL
  – CRTMQMCHL
- Inclusion of channel auto definitions.

  Addition of CHAD, CHADEV, and CHADEXIT fields to the CHGMQM command.
- Minor technical and editorial improvements throughout the book.

Part 1. Guidance

© Copyright IBM Corp. 1994, 1998

Chapter 1. Introduction

This chapter introduces the IBM MQSeries for AS/400 product from an administrator's perspective, and describes the basic concepts of MQSeries for AS/400 and messaging.

MQSeries for AS/400 and message queuing

MQSeries for AS/400 lets OS/400 applications use message queuing to participate in message-driven processing. With message-driven processing, applications can communicate across different platforms by using the appropriate message queuing software products. For example, OS/400 and MVS/ESA applications can communicate through MQSeries for AS/400 and MQSeries for MVS/ESA respectively.

MQSeries products implement a common application programming interface (Message Queue Interface or MQI) whatever platform the applications run on. The calls made by the applications and the messages they exchange are common.

Time-independent applications

With message queuing, the exchange of messages between the sending and receiving programs is time independent. This means that the sending and receiving applications are decoupled so that the sender can continue processing without having to wait for the receiver to acknowledge the receipt of the message.

Event-driven processing

Applications can be started by messages arriving on a queue and, if necessary, terminated when the message or messages have been processed.

Messages and queues

Messages and queues are the basic components of any queuing system.

What is a message?

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

In MQSeries for AS/400, messages have two parts: the *application data* and a *message descriptor*. The content and structure of the application data is defined by the application programs that use them. The message descriptor identifies the message and contains other control information, for example, the type of message, and the priority assigned to the message by the sending application.

The format of the message descriptor is defined by MQSeries for AS/400. For a complete description of the message descriptor, see the *MQSeries Application Programming Reference* manual or the *MQSeries for AS/400 Application Programming Reference (RPG)*, as appropriate.

© Copyright IBM Corp. 1994, 1998

All applications that participate in message queuing use messages that conform to this common specification. In MQSeries for AS/400, there are four types of message:

Datagram	A simple message for which no reply is expected.
Request	A message for which a reply is expected.
Reply	A reply to a request message.
Report	A message that describes an event such as the occurrence of an error.

The message type is part of the message descriptor.

Message lengths

In MQSeries for AS/400, the maximum message length is 4 MB where MB equals 1 048 576 bytes. In practice, the message length may be 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 applications, when one of the applications is operating in a non-MQSeries for AS/400 platform.
- The amount of storage available for the message.

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

What is a queue?

In physical terms, a *queue* is a type of list that is used to store messages until they are retrieved by an application.

Queues exist independently of the applications that use them. A queue can exist in main storage, if it is temporary; on disk or similar auxiliary storage, if it must be kept in case of recovery; or in both places, if it is currently being used, and must also be kept for recovery. Each queue belongs to a *queue manager*, which is responsible for maintaining it. The queue manager puts the messages it receives onto the appropriate queue.

Queues can exist either in your local system, where they are called *local queues*, or at another queue manager, where they are called *remote queues*.

In MQSeries, messages can be retrieved from a queue by suitably authorized applications according to these retrieval algorithms:

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

For more information about queues and their attributes, refer to the *MQSeries Application Programming Guide*.

Objects and commands

Many of the tasks described in this book involve manipulating MQSeries for AS/400 *objects*, of which, there are four different types:

1. Message queue managers
2. Queues, subdivided into:
   - Local queues
   - Remote queues
   - Alias queues
   - Model queues
3. Process definitions
4. Channels, subdivided into:
   - Receiver channels
   - Sender channels
   - Server channels
   - Server-connection channels
   - Requester channels

Object names

Each message queue manager is identified by a unique name.

For the other types of objects, each object has a name associated with it and can be referenced in MQSeries for AS/400 commands by that name. Names must be unique within each of these object types. 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 MQSeries for AS/400, names can have a maximum of 48 characters, with the exception of channels which can have a maximum of 20 characters.

MQSeries for AS/400 queue manager

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

Note: The AS/400 machine only supports one queue manager, whereas other platforms support multiple queue managers.

To an application a queue is a *local queue* if it is managed by the same message queue manager that is connected to the application. If the queue is managed by a different message queue manager, it is called a *remote queue*.

Message queue manager objects

A message queue manager provides queuing services to applications and commands to administrators, but it is also an MQSeries for AS/400 object in its own right. Because it is an object, a message queue manager may be used in both MQI calls and MQSeries for AS/400 commands.

MQI calls: A queue manager object may be used in some MQI calls. For example, you can inquire about the attributes of the message queue manager object using the MQI call MQINQ. However, you cannot put messages on a queue manager object; messages are always put on queue objects, not to message queue manager objects.

MQSeries for AS/400 queues

Queues are defined to MQSeries for AS/400 using the CRTMQMQ command and specifying the type of queue, or by using the STRMQMMQSC command and supplying the creation details in a command file. These commands specify the attributes of the queue being defined. 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 “CRTMQMQ (Create MQM Queue) Command” on page 168.

Using queue objects

In MQSeries for AS/400, there are four types of queue object. Each type of object can be manipulated by MQSeries for AS/400 commands and is associated with real queues in different ways:

1. A *local queue* object identifies a local queue belonging to the queue manager to which the application is connected. Messages always end up on a queue that is local to a queue manager.
2. A *remote queue* object identifies a queue belonging to another queue manager. This queue, therefore, must be defined as a local queue to that queue manager. The information you specify when you define a remote queue object allows the queue manager to find the remote queue manager, so that any messages destined for the remote queue go to the correct queue manager.
3. An *alias queue* object 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 merely 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 you can use to access another queue.
4. A *model queue* object 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 may be temporary queues, which do not survive product restarts, or permanent queues, which do.

Use the CRTMQMQ command to create one of these types of queue objects. You can also use the default queue objects supplied with MQSeries for AS/400 as the basis of your definitions. The sample program that defines these default objects is described in "AMQSDEF4" on page 241.

Specific local queues used by MQSeries

MQSeries for AS/400 uses some local queues for specific purposes related to its operation. These queues must be defined before MQSeries for AS/400 can use them.

The MQSeries for AS/400 administrator is responsible for defining and maintaining all queues using the information in this book.

Initiation queues

The message queue manager puts trigger messages on an *initiation queue* when applications use the triggering feature to start programs automatically. Typically, a trigger event occurs when a message is put on a queue so that the trigger conditions for that queue are met. For example, the trigger conditions could be that a trigger event occurs when the number of messages on the queue reaches a predefined depth. The trigger event causes the queue manager to put a trigger message on the initiation queue for that queue. This trigger message is retrieved by a trigger monitor application. The trigger monitor then starts up the application program that was specified in the trigger message.

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

A *trigger monitor* is an application that monitors an initiation queue. When a trigger message arrives on the initiation queue, it is retrieved by the trigger monitor. Typically, the trigger monitor then starts an application that is specified in the message.

Transmission queues

A *transmission queue* 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. For details about the use of transmission queues in distributed queuing, see the *MQSeries Intercommunication* book.

Event queues

An *event queue* is a queue that is used to receive *event messages*, which indicate that a particular type of instrumentation event has occurred during the execution of an application program. There are three system administration event queues, one for each of the three categories of instrumentation event that can be generated:

- SYSTEM.ADMIN.QMGR.EVENT - for queue manager events
- SYSTEM.ADMIN.PERFM.EVENT - for performance events
- SYSTEM.ADMIN.CHANNEL.EVENT - for channel events

When an event is generated, it is put on one of these queues. See the *MQSeries Programmable System Management* manual for more information about events.

Undelivered-message queue

An *undelivered-message (dead-letter) queue* receives messages that cannot be routed to their correct destinations. This occurs when, for example, the destination queue is full.

For distributed queuing, you should define an undelivered-message queue for each queue manager involved.

SYSTEM.ADMIN.COMMAND.QUEUE

The *SYSTEM.ADMIN.COMMAND.QUEUE* is a local queue to which suitably authorized applications can send MQSeries for AS/400 commands. These commands are then retrieved by an MQSeries for AS/400 component called the command server. The command server validates the commands and passes the valid ones on for processing by the MQSeries for AS/400 command processor.

A SYSTEM.ADMIN.COMMAND.QUEUE must be defined for each queue manager. A sample SYSTEM.ADMIN.COMMAND.QUEUE is defined in the sample program AMQSDEF4, the source of which is shown in “AMQSDEF4” on page 241.

System default queues

The *system default queues* are a set of queue definitions supplied with MQSeries for AS/400. You can copy and rename any of these queue definitions for use in applications at your installation.

For example, to define a local queue, you can copy the supplied default SYSTEM.DEFAULT.LOCAL.QUEUE, change its name, and then alter any of its other attributes, as required.

Process definitions

A *process definition object* defines an application that is to be started in response to a trigger event on an MQSeries for AS/400 queue manager. In MQSeries for AS/400, an application retrieves messages from one or more specified queues and processes them.

This definition includes the application ID, the application type, and data specific to the application.

Use the CRTMQMPRC command to create a process definition. You can also use the default process definition object, SYSTEM.DEFAULT.PROCESS (supplied with MQSeries for AS/400) as the basis of your own definitions.

Channels

A *channel* is used in distributed message queuing to move messages from one queue manager to another. For information on channels and how to use them, see the *MQSeries Intercommunication* book.

A channel is also used to move messages between an *MQSeries client* and an *MQSeries server*. For information about the channels used with MQSeries clients and servers, see the *MQSeries Clients* book.

| Message channels can be customized using an initialization file called QMINI. A
| default version is created in library QMQMDATA when MQSeries is installed. For
| information about the initialization parameters, see the *MQSeries
| Intercommunication* book.

MQSeries clients with MQSeries for AS/400 V4R2M1 as a server

MQSeries for AS/400 V4R2M1 includes support for the attachment of MQSeries clients on other platforms.

An MQSeries client is a component of the MQSeries product that can be installed on its own, on a separate machine from the full queue manager and server. You can run an MQSeries application on an MQSeries client and it can interact, by means of a communication protocol, with one or more MQSeries servers and connect to their queue managers.

MQSeries for AS/400 can be used as an MQSeries server, but not as an MQSeries client.

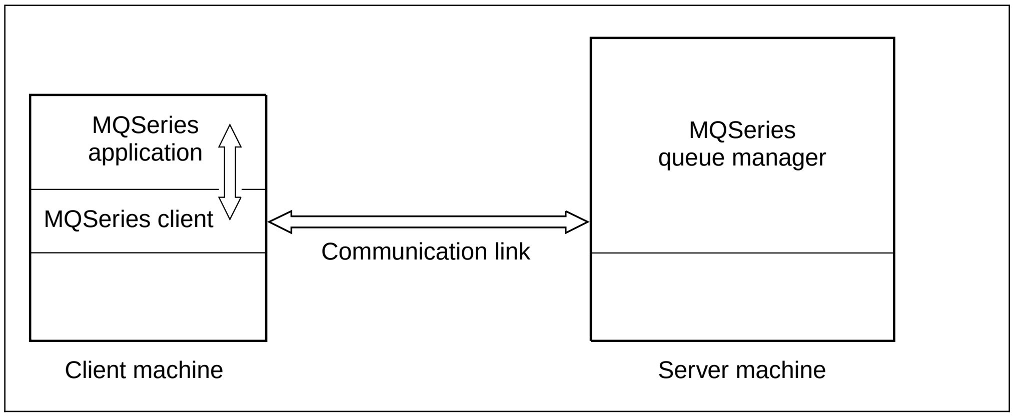


Figure 1. Client / server relationship

The communication link has to be set up and configured first. Then the *MQI channel*, the logical communication link between the MQSeries client and server, has to be defined at each end. Use the channel type (CHLTYPE) of Server connection (*SVRCN) in the CRTMQMCHL (Create MQM Channel) command to create the MQI channel definition at the server end on MQSeries for AS/400.

For information about configuring the communication links and defining the MQI channels on the MQSeries client, see the book *MQSeries Clients* .

- Although MQSeries clients cannot be run on an AS/400 system, you can connect MQSeries clients on other platforms to MQSeries for AS/400 V4R2M1.
- Files for MQSeries clients are ***not*** supplied with MQSeries for AS/400 V4R2M1.
- If you have MQSeries for AS/400 only, you can get the MQSeries client files from SupportPacs.

MQSeries clients from IBM Transaction Processing SupportPacs

The MQSeries client files can be copied from the IBM Transaction Processing SupportPacs for use as needed.

The IBM Transaction Processing SupportPacs library consists of material that complements the family of CICS and MQSeries products marketed by IBM. The Transaction Processing SupportPacs library is available on the Internet at:

```
http://www.software.ibm.com/ts/mqseries/
```

MQSeries client software is available at no charge, and is subject to the IPLA and License Information terms defined when requesting the MQSeries clients on the Internet. You have the right to make as many copies of the MQSeries client as necessary.

Chapter 2. Installing and migrating MQSeries for AS/400

This chapter describes how to install and migrate MQSeries for AS/400.

Note to users:

| You **must** have the OS/400 V4R2 or V4R3 operating system installed on your
| machine in order to run MQSeries for AS/400 V4R2M1. If you have the OS/400
| V4R1 or V3R7 operating system installed, you can use only MQSeries for OS/400
| V3R6 or MQSeries for OS/400 V3R7.

MQSeries for AS/400 – installable options

The options are:

MQSeries for AS/400
The base product

MQSeries for AS/400 - Samples
Sample resource definitions

MQSeries for AS/400 - Admin Application
The MQSeries for AS/400 Administration utility, which enables you to use a panel interface to the queue manager.

Notes:

1. The Administration utility need not be loaded if you are installing MQSeries for AS/400 for remote use only, that is, at a site where there is no system administrator.
2. There can be only one instance of MQSeries for AS/400 on each AS/400 machine. MQSeries must not be running during the installation process.

Setting system values

Before installing MQSeries for AS/400, you should check – using the DSPSYSVAL command – that the following system values are set to the requirements of your enterprise:

- QCCSID
- QUTCOFFSET
- QSYSLIBL
- QALWOBJRST

You can change these values, if necessary, using the CHGSYSVAL command.

QCCSID

All messages have the CCSID as a tag in the header, which identifies the codepage and character set of the source. For CCSIDs supported on the AS/400 see the *AS/400 National Language Support* book.

The CCSID information is obtained, when the queue manager is created, from the job CCSID.

If this does not contain a valid value of 1-65534, the information is obtained from the default CCSID value for the job.

Note: The CCSID must be:

- Single byte character set (SBCS), or
- Mixed, that is SBCS and DBCS.

If the job CCSID is DBCS only, the CRTMQM command fails.

QUTCOFFSET

You should check that the coordinated universal time offset (QUTCOFFSET) system value has been set, to indicate the relationship between the system time and Greenwich Mean Time (GMT). You do this by working with the CHGSYSVAL command.

If QUTCOFFSET is not set, it takes the default value of zero. MQSeries for AS/400 then assumes that the local system time is universal time coordinated (UTC), that is GMT, and timestamps the MQSeries for AS/400 messages accordingly.

QSYSLIBL

Ensure that QSYS2 is included in the list of libraries that make up the system part of the library list.

MQSeries for AS/400 uses programs in this library for data conversion and SNA LU 6.2 communication.

QALWOBJRST

QALWOBJRST needs to include QMQMDATA and QMQMPROC during installation. This can be done specifically, or by using the system value *ALL.

If the system value is set to *NONE, installation fails.

You should reset QALWOBJRST to its original value to maintain system security.

Before you install

If you are migrating from an earlier release of MQSeries for AS/400, see “Migrating to V4R2M1” on page 18.

In order to use the standard AS/400 installation process, the product identifier for the version of MQSeries for AS/400 you are installing **must** match exactly that listed, which is 5769-MQ2. To check this, carry out the following procedure:

1. Type GO LICPGM on the command line.
2. Select option 11 (Install licensed program).
3. Find the entry for MQSeries for AS/400 in the list of products.

If the product identifier listed **is** the same as the product identifier of the version of MQSeries for AS/400 that you are installing, follow the standard procedure in “Standard AS/400 installation procedure” on page 15.

If the product identifier listed is **not** the same as the product identifier of the version of MQSeries for AS/400 that you are installing, you **must** use the following commands instead of the menu interface:

1. To install MQSeries for AS/400, issue the command:

   ```
   RSTLICPGM LICPGM(product identifier) DEV(tape device)
   ```

   where:

   - `product identifier` is the product identifier for the version of MQSeries for AS/400 that you wish to install.
   - `tape device` is the device containing the tape from which the product should be loaded.

2. To install the MQSeries for AS/400 samples, issue the command:

   ```
   RSTLICPGM LICPGM(product identifier) DEV(tape device) OPTION(1)
   ```

   where:

   - `product identifier` is the product identifier for the version of MQSeries for AS/400 that you wish to install.
   - `tape device` is the device containing the tape from which the product should be loaded.

3. To install the MQSeries for AS/400 Administration Application issue the command:

   ```
   RSTLICPGM LICPGM(product identifier) DEV(tape device) OPTION(2)
   ```

   where:

   - `product identifier` is the product identifier for the version of MQSeries for AS/400 that you wish to install.
   - `tape device` is the device containing the tape from which the product should be loaded.

Standard AS/400 installation procedure

To install MQSeries for AS/400 use the standard AS/400 install process:

1. Type GO LICPGM on the command line
2. Select option `11` (Install licensed program)
3. Select the installable options, for MQSeries for AS/400 that you require

It takes approximately 10 minutes to install the MQSeries for AS/400 base product, and both options, on an AS/400 Model 510 (from a tape type 6525, density *QIC 525).

This time has been recorded when the AS/400 system concerned is performing no other work.

Verifying the product installation

To ensure that the tape has loaded correctly, carry out the following procedure:

1. Use GO LICPGM with option 10 and check that licensed program 5769MQ2 displays status `*COMPATIBLE`.

   Note: When verifying the product installation for MQSeries for AS/400 V4R2M1 on operating system V4R2, the description field displays the following:

   `5769MQ2 00`, for the Base option

   `5769MQ2 01`, for the installed Samples option

   `5769MQ2 02`, for the installed Administration Application option

2. Install any PTFs that are required.

If the installed status is **not** `*COMPATIBLE`, you must delete the product and install it again.

Quiescing MQSeries for AS/400

The orderly shutdown of MQSeries for AS/400 is called *quiescing*. You may need to quiesce MQSeries for AS/400, for example, to:

- Take a backup of the system, or
- Update MQSeries for AS/400

To quiesce the system, you should:

1. Use `F12` (Cancel) to return to your initial MENU

   Note: If you have MQSeries Commands (CMDMQM) as your initial menu, change the initial menu in your user profile, sign off and sign back on.

2. Ensure that you have:

   - `*ALLOBJ` authority, or object management authority for the `QMQM` and `QMQMADM` libraries.
   - *USE authority for either, or both, `QMQM/AMQIQES4` and `QMQM/AMQIQEJ4`.

   Notes:

   a. If you use `QMQM/AMQIQES4` any user signed on to MQSeries, even if they are not running the product, will be logged off.

   b. The `QMQMADM` library exists only if you have installed the MQSeries for AS/400 Administration utility.

3. Warn all users that you are going to stop MQSeries for AS/400.
4. Warn users not to start Message Queue Interface applications while MQSeries for AS/400 is being quiesced. To ensure that they cannot start, you should revoke users' authorities to MQSeries for AS/400 by issuing the RVKMQMAUT command.
5. Optionally, issue the command ENDMQM with the *OPTION* parameter set to `*CNTRLD`.

This prevents new applications connecting to MQSeries for AS/400, but allows applications to complete any work already started.

6. Exit the Administration utility, if it is running.

When you have carried out the preceding steps you can end the queue manager in one of two ways:

1. Issue the command ENDMQM with the *OPTION* parameter set to `*IMMED`, followed by `CALL QMQM/AMQIQES4`.

   The AMQIQES4 program:

   - Ends all jobs or user sign ons that have a lock on the MQSeries QMQM library object. The exception is the calling sign on, in which case the library QMQM is removed from the library list.
   - Ends those jobs that are licensed users of MQSeries.
   - Revokes all object authorities to libraries `QMQM`, and `QMQMADM` for LICPGM option two.

   This program is designed to be used prior to:

   - A powerdown of the operating system
   - Installing PTFs
   - Deleting, or installing, the MQSeries licensed program.

2. Issue the command `CALL QMQM/AMQIQEJ4`.

   This program:

   - Ends those jobs owned by `QMQM` only.
   - Does ***not*** terminally revoke the object authorities to libraries `QMQM` and `QMQMADM`.
   - Performs a Record Object Image (RCDMQMIMG) operation before ending the queue manager.

     Note: This happens only if you have not already issued the command ENDMQM.

   This program is designed to be used to shutdown MQSeries:

   - At the end of the day
   - Prior to taking backups of the MQSeries journals and MQSeries data.

Notes:

1. If you have quiesced MQSeries for AS/400 using the program AMQIES4, you are recommended to issue the command `CALL QMQM/AMQIQEM4`. This program releases the user space associated with MQSeries for AS/400.
2. If you are going to reinstall MQSeries, follow the instructions given in "Migrating to V4R2M1" on page 18.
3. When you install, or reinstall, MQSeries for AS/400 grants `*PUBLIC *USE` authority to the `QMQM` and, if the Administration utility is loaded, the `QMQMADM` libraries.

   Alternatively, you can use the GRTOBJAUT command to grant `*PUBLIC *USE` authority to the `QMQM` and, if the Administration utility is loaded, the `QMQMADM` libraries, if you have quiesced MQSeries for AS/400 using AMQIQES4, and are not installing or reinstalling.

Deleting MQSeries for AS/400 V4R2M1

To delete MQSeries for AS/400 you:

1. Quiesce the product. See “Quiescing MQSeries for AS/400” on page 16.
2. Use the GO LICPGM menu and select option 12 “Delete licensed program.”
3. Select the installable options of MQSeries for AS/400 to be deleted.

Notes:

1. Deleting the *BASE option deletes *ALL* the objects in the QMQMDATA and QMQMPROC libraries. Therefore, your enterprise should not create objects in these libraries.
2. Delete the *BASE option, only if you do not want to save any data for when you reinstall MQSeries for AS/400.
3. Certain objects still exist after the queue manager is disconnected. A list of objects and jobs created by MQSeries for AS/400 is given in “Objects created by MQSeries for AS/400” on page 275. The objects will be removed when you sign off.

Migrating to V4R2M1

There are two major types of MQSeries upgrade:

1. The AS/400 upgrade takes place on the same machine, and may optionally be accompanied by a hardware upgrade. This is sometimes referred to as a “Slip-install”.

   For this type of upgrade there are three classes available:

   Product code only
   The MQSeries licensed program is deleted, which deletes both the code and data libraries. This is followed by the installation of the new version of MQSeries.

   Product code and definitions
   You can do this in two ways:

   - As for product code only, after saving MQ object definitions in:
     – A CL program, similar to the source for QMQM/AMQSDEF4
     – An MQSC program, similar to member AMQSCOMA in QMQMSAMP/QMQSC

     This method allows the queue manager name to be changed during the upgrade, using DLTMQM and CRTMQM.
   - As for product code, definitions and message data, without specifically saving the MQSeries journals. Note that this method ***must*** inherit the original queue manager name.

   Product code, definitions, and message data
   MQSeries is reinstalled over the top of the earlier version, without deleting any of the data libraries (QMQMDATA and QMQMPROC) or the MQSeries journals in library QUSRSYS.
2. The AS/400 upgrade takes place on a different machine. This is sometimes referred to as a “Side-by-side install”.

For this type, only the **Product code, definitions, and message data** class is described.

Preparing for migration

The first thing that you need to do is prepare your source machine:

1. End all MQSeries channels by using WRKMQMCHL and option 15.
2. Exit the Administration utility if it is running,
3. End the MQSeries Command Server by using the command:

   ```
   ENDMQMCSVR MQMNAME(FRED) OPTION(*IMMED)
   ```

   where FRED is the name of the queue manager.
4. Reset the media recovery point by using the command:

   ```
   RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL)
   ```
5. Quiesce the product by following the instructions given in "Quiescing MQSeries for AS/400" on page 16.
6. Copy all the definition files that you need into QGPL/QMQSC, or QGPL/QCLSRC, or both.
7. For an MQSeries upgrade:
   a. Issue the command CRTSAVF FILE(QGPL/QMQMxxxx), where xxxx represents DATA and PROC.
   b. Save the libraries QMQMDATA and QMQMPROC into the save files that you have created, using the command SAVLIB LIB(QMQMxxxx) DEV(*SAVF) SAVF(QGPL/QMQMxxxx).
   c. Issue the command CRTSAVF FILE(QGPL/QMQMJRN).
   d. Save the journals into the save file just created, using the command SAVOBJ OBJ(AMQ*) LIB(QUSRSYS) DEV(*SAVF) OBJTYPE(*JRN) SAVF(QGPL/QMQMJRN).

   Notes:

   a. Ensure that you have backup copies of the MQSeries save files and data definitions that you have saved into library QGPL.
   b. It should not be necessary to specify a target release; the default value *CURRENT allows objects to be restored to a later release.
8. Upgrade your AS/400, and optionally upgrade your hardware, using the appropriate AS/400 books.

Installation on the same machine

The three types of upgrade are described in the following sections.

Product code only upgrade

On the machine:

1. Type GO LICPGM on the command line.
2. Select option 12 (Delete licensed programs).
3. Select the options of MQSeries for AS/400 to be deleted.
4. Select option 11 (Install licensed program). See "Before you install" on page 14 for further information.

5. Select the installable options, for MQSeries for AS/400 that you require.
6. Apply all PTFs that are recommended in the appropriate PSP database on RETAIN.

This completes the installation method for this class of upgrade on the same machine and you should now verify that MQSeries has been installed correctly. See “Verifying the installation” on page 32 for further details.

Product code and definitions upgrade

On the machine:

1. Type GO LICPGM on the command line.
2. Select option 12 (Delete licensed programs).
3. Select the options of MQSeries for AS/400 to be deleted.
4. Select option `11` (Install licensed program). See “Before you install” on page 14 for further information.
5. Select the part, or parts, for MQSeries for AS/400 that you require.
6. Apply all PTFs that are recommended in the appropriate PSP database on RETAIN.
7. Start MQSeries using STRMQM.
8. Refresh the MQSeries default objects by issuing the command `CALL QMQM/AMQSDEF`.
9. Replace the definitions for your objects by running your MQSC or CL commands, or both, saved in the QGPL/QMQSC or QGPL/QCLSRC files respectively.

This completes the installation method for this class of upgrade on the same machine and you should now verify that MQSeries has been installed correctly. See “Verifying the installation” on page 32 for further details.

Product code, definitions, and message data upgrade

On the machine:

1. Type GO LICPGM on the command line.
2. Select option `11` (Install licensed program). See “Before you install” on page 14 for further information.
3. Select the installable options, for MQSeries for AS/400 that you want to reinstall.
4. Apply all PTFs that are recommended in the appropriate PSP database on RETAIN.

This completes the installation method for this class of upgrade on the same machine and you should now check that MQSeries is working on the machine. See “Verifying the upgrade” on page 22 for further details.

Installation on another machine

If you want to restore the current queue manager and its contents on the target machine, take a backup of libraries QMQMDATA, QMQMPROC and the MQSeries journals in library QUSRSYS (objects AMQ* of type *JRN).

On the target machine:

1. Type GO LICPGM on the command line.
2. Select option 12 (Delete licensed programs).
3. Select the options of MQSeries for AS/400 to be deleted.
4. Restore the MQSeries objects from the source machine, for libraries QMQMDATA and QMQMPROC using the following commands:

   ```
   RSTLIB SAVLIB(QMQMDATA) DEV(*SAVF) SAVF(QGPL/QMQMDATA)
   RSTLIB SAVLIB(QMQMPROC) DEV(*SAVF) SAVF(QGPL/QMQMPROC)
   ```

5. Restore the MQSeries local and remote journals from the source machine using the following command:

   ```
   RSTOBJ OBJ(AMQ*) SAVLIB(QUSRSYS) DEV(*SAVF) OBJTYPE(*JRN)
             SAVF(QGPL/QMQMJRN)
   ```

6. Reinstall the required version of MQSeries.

   Note: MQSeries install updates all existing channel definitions (in file QMQMDATA/AMQRFCD4) to match the structures required by the later target release.

7. After you have installed the required version of MQSeries you need to associate the journal receivers. To do this:

   a. Issue the command WRKJRN, specifying the journal AMQAJRN.
   b. Use option 9 to associate the journal.

8. Apply all PTFs that are recommended in the appropriate PSP database on RETAIN.

This completes the installation method on another machine and you should now check that MQSeries is working on the machine. See “Verifying the upgrade” on page 22 for further details.

Channel definitions

If you do not have any channels defined, and you migrate to a later version of MQSeries for AS/400, you may not be able to create a channel.

This can occur if:

- You install MQSeries without creating the system default objects correctly. For example, you have not run `CALL QMQM/AMQSDEF4`.
- You then migrate to a later release of MQSeries for AS/400.

You can overcome this problem by:

1. Quiescing MQSeries for AS/400 – see “Quiescing MQSeries for AS/400” on page 16.
2. Removing the existing empty channel definition file, using the following command:

| ```
DLTF FILE(QMQMDATA/AMQRFCD4)
```

| 3. Creating the new empty channel definition file, using the following command:

| ```
CRTPF FILE(QMQMDATA/AMQRFCD4) SRCFILE(QMQM/QDDSSRC)
```

| 4. Restarting MQSeries for AS/400.

Verifying the upgrade

Use the following procedure to check that you have installed the **Product code, definitions and message data upgrade** successfully:

1. Ensure that ***all*** users have access to ***all*** MQSeries libraries, QMQM, QMQMDATA, and QMQMPROC, plus QMQMSAMP (if LICPGM option 1 is installed) and QMQMADM (if LICPGM option 2 is installed).

   For example:

   ```
   GRTOBJAUT OBJ(NNNNN) OBJTYPE(*LIB) USER(*PUBLIC) AUT(*EXECUTE)
   ```

   where NNNNN is the name of an MQSeries library.

2. Start MQSeries, using STRMQM.

3. Refresh MQSeries default object attributes by issuing the command:

   ```
   CALL QMQM/AMQSDEF4
   ```

4. If applicable, verify that the queue manager and its objects have been successfully retained.

   Example commands are:

DSPMQM	Display Queue Manager
DSPMQMOBJN OBJ(*ALL)	Display object names
WRKMQMCHL	Display list of channels
WRKMQMMSG	Display persistent messages

5. Verify that the Administration utility runs successfully. See "Verifying the Administration utility" on page 46.

6. Use commands DSPOBJAUT or DSPMQMAUT to review your authorizations to all types of MQSeries objects (*MQM, *CTLG, *ADM, *Q, *ALSQ, *LCLQ, *RMTQ, *PRC, *CMD).

   See "MQSeries authorities" on page 25 for further details.

Chapter 3. Security

This chapter introduces aspects of security associated with MQSeries for AS/400.

MQSeries for AS/400 uses OS/400 system security when accessing system objects and you should be familiar with the *AS/400 Security - Basic* and *AS/400 Security - Advanced* books. However, as a member of the MQSeries family of products, the security accesses and controls are slightly different from other AS/400 operations.

This chapter describes the differences and shows how the two systems are mapped together.

Introduction

MQSeries for AS/400 provides commands to grant, revoke, and query the authority that an application, or user, has to do the following:

- Issue MQSeries for AS/400 commands
- Perform operations on MQSeries for AS/400 objects

The OS/400 operating system has a large quantity of object access information associated with each system object. Access to objects is enforced by the system based on this information. To create, delete, open, modify, and perform other operations on an object, applications must have the authorities to perform these tasks listed in the access information tables.

MQSeries for AS/400 security considerations

MQSeries for AS/400 has no specific requirements concerning security. The following items are included, for information, so that you can ensure that the security of your enterprise is maintained:

1. You should grant and revoke authorities to the MQSeries for AS/400 commands using the AS/400 GRTOBJAUT and RVKOBJAUT commands.
2. When installed, MQSeries for AS/400 sets up a special user profile `QMQM`, which cannot have a password allocated to it. MQSeries for AS/400 libraries and objects are owned by this user profile. Any Message Queue Interface (MQI) calls adopt the authority of `QMQM` and perform the necessary MQSeries for AS/400 object authority checks on behalf of the caller.

   However, this user profile can be used in certain circumstances. For example, you can use the AS/400 WRKSPLF command and select the option to display data in the spooled file, for user `QMQM`, if a channel fails to start.
3. You should leave the MQSeries for AS/400 libraries `QMQMDATA` and `QMQMPROC` with the authority of `PUBLIC(*EXECUTE)`.
4. The system value for QALWUSRDMN (Allow User Domain Objects) *requires* the inclusion of the libraries `QMQMDATA` and `QMQMPROC`.
5. You should allocate privileged users the appropriate authorities to individual MQM objects, using the GRTMQMAUT command.

Note: It is important that you use the appropriate MQSeries for AS/400 commands to grant or revoke MQSeries object authorities, and not the standard AS/400 commands, because MQSeries for AS/400 object names (which can have a maximum of 48 characters for objects other than channels and 20 characters for channels) may be different from the names of the AS/400 objects that represent them.

6. You need to grant authority to create queues.

   In order to prevent the security of the system being circumvented, this authority should be limited to privileged users. This is because, when opening an alias queue, the MQI performs the authorization checking against the alias queue only, and not the queue that it addresses.

7. You should 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)

8. You should limit the number of users who have authority to work with special queues that are particularly sensitive. These queues include:

   - Transmission queues
   - The SYSTEM.ADMIN.COMMAND.QUEUE

9. The trace commands require special considerations; see “Security considerations for tracing MQSeries for AS/400” on page 25.

10. You should ensure that particular care is taken with applications that use the full MQI message context options.

11. Channel definitions contain a security exit program specification. Channel creation and modification requires special considerations; see “Security considerations for creating and changing channels” on page 28. Details of security, concerning exits, are given in the *MQSeries Intercommunication* book.

12. You need to be aware that the channel exit and trigger monitor programs can be substituted. The security of such replacements is the responsibility of the programmer.

13. The following authority commands only work for authorized users of the queue manager:

    DSPMQMAUT (Display MQM Authority)
    GRTMQMAUT (Grant MQM Authority)
    RVKMQMAUT (Revoke MQM Authority)

14. QSYS must have the special authority *ALLOBJ.

Security considerations for tracing MQSeries for AS/400

In addition to the standard AS/400 trace facilities available, MQSeries for AS/400 uses the following trace commands:

- STRMQMSRV
- ENDMQMSRV
- TRCMQM

To use the STRMQMSRV or ENDMQMSRV commands, you must have *ALLOBJ authority, or be signed on to the system with one of the following user IDs:

- QPGMR
- QSYSOPR
- QSRV
- QSRVBAS

The TRCMQM command is shipped with PUBLIC *EXCLUDE authority, and with the following user profiles having private authorities to use the command.

- QPGMR
- QRJE
- QSYSOPR
- QSRV
- QSRVBAS

MQSeries authorities

Access to MQSeries objects is controlled by authorities to:

1. Issue the MQSeries command
2. Access the MQSeries objects referenced by the command

Note: Depending on the type of command issued, there may be additional checking of the MQM Catalog and the MQM Administration object.

Granting MQSeries authorities to MQSeries commands

You should grant and revoke authorities to the MQSeries for AS/400 commands using the AS/400 GRTOBJAUT and RVKOBJAUT commands.

Granting MQSeries authorities to MQSeries objects

1. You must either grant authority to MQSeries for AS/400 objects using the GRTMQMAUT command, or revoke the authority to MQSeries for AS/400 objects using the RVKMQMAUT command.
2. The authority can be granted in one of two ways, by using:
   - AS/400-specific authority values (for example, *READ and *OBJOPR) or,
   - MQSeries for AS/400 authority values (for example, *MQMPASSALL)
3. To display MQSeries for AS/400 authority on the objects, use the DSPMQMAUT command.

Message Queue Manager catalog object

MQSeries for AS/400 maintains a mapping between the 48-character names of MQSeries for AS/400 objects and the 10-character names of the operating system objects that represent them. An MQSeries for AS/400 object with the same name as the queue manager, and with object type `*CTLG`, is a catalog holding the mapping, and also the authorities to create or delete MQSeries for AS/400 objects.

Message Queue Manager administration object

An MQSeries for AS/400 object with the same name as the queue manager, and with object type `*ADM` (called Message Queue Manager administration object), has the sole purpose of supporting the authorities to pass or set message context, or to specify Alternate User Id authority checking. An application specifying one of these options needs authority, as shown in Table 3 on page 27 and Table 4 on page 28.

Authorities needed to run an MQSeries command

To run a command requires the following authorities:

1. Authority to the command (see "Default command authorities" on page 29).
2. Authority to the objects that the command references.
3. Authority to the MQSeries for AS/400 catalog object (`*CTLG`).

Table 1 (Page 1 of 2). Authorities for operation and control commands

MQSeries for AS/400 Command	**MQSeries object (*OBJ) authority required - use GRTMQMAUT**	**MQSeries catalog (*CTLG) authority required - use GRTMQMAUT**
Changing MQSeries for AS/400 objects		
CHGMQM CHGMQMQ CHGMQMPRC	*READ and *UPD	n/a
Clearing MQSeries for AS/400 queue objects		
CLRMQMQ	*READ and *DLT	n/a
Copying without replacement of existing MQSeries for AS/400 objects		
CPYMQMQ CPYMQMPRC	*READ authority on object being copied	*ADD
Copying with replacement of existing MQSeries for AS/400 objects		
CPYMQMQ CPYMQMPRC	*READ authority on object being copied *OBJOPR and *UPD authority on object being replaced	n/a
Creating without replacement of existing MQSeries for AS/400 objects		
CRTMQMQ CRTMQMPRC	*READ authority on system default object	*ADD
Creating with replacement of existing MQSeries for AS/400 objects		
CRTMQMQ CRTMQMPRC	*READ authority on system default object *OBJOPR and *UPD authority on object being replaced	n/a
Deleting MQSeries for AS/400 objects		
DLTMQMQ DLTMQMPRC	*OBJEXIST	*DLT
Displaying MQSeries for AS/400 objects		
DSPMQM DSPMQMQ DSPMQMPRC	*READ	n/a

Table 1 (Page 2 of 2). Authorities for operation and control commands

MQSeries for AS/400 Command	MQSeries object (*OBJ) authority required - use GRTMQMAUT	MQSeries catalog (*CTLG) authority required - use GRTMQMAUT
Granting and revoking authority to MQSeries for AS/400 objects		
GRTMQMAUT RVKMQMAUT	*OBJMGT	n/a

Authority checking for the MQI

For the MQI, the authority is performed on the MQOPEN call and not on each MQGET, MQPUT, and so on.

Table 2 gives the authorities required to open an MQSeries object.

Table 3 and Table 4 on page 28 give the authorities necessary when opening an MQSeries object to pass or set message context, or to specify Alternate User Id.

Table 2. Authorities required to open an object

Opening MQSeries object with option MQOO_	MQSeries object (*OBJ) authority required - use GRTMQMAUT	MQSeries object authority shown - use DSPMQMAUT										
		Authority	Object					Data				
			Opr	Mgt	Exist	Alter	Ref	Read	Add	Upd	Del	Execute
BROWSE	*READ	USER DEF	-	-	-	-	-	√	-	-	-	-
INPUT_AS_Q_DEF	*READ and *DLT	USER DEF	-	-	-	-	-	√	-	-	√	-
INPUT_EXCLUSIVE	*READ and *DLT	USER DEF	-	-	-	-	-	√	-	-	√	-
INPUT_SHARED	*READ and *DLT	USER DEF	-	-	-	-	-	√	-	-	√	-
INQUIRE	*READ	USER DEF	-	-	-	-	-	√	-	-	-	-
OUTPUT	*ADD	USER DEF	-	-	-	-	-	-	√	-	-	-
SET	*UPD	USER DEF	-	-	-	-	-	-	-	√	-	-

Table 3. Authorities for Context, AlternateUserID - MQSeries for AS/400 details

Opening MQSeries object with option MQOO_	MQSeries object (*OBJ) authority required - use GRTMQMAUT	MQSeries object authority shown - use DSPMQMAUT					
		Authority	PassId	PassAll	SetId	SetAll	AltUsr
ALTERNATE_USER AUTHORITY	*DLT	USER DEF	-	-	-	-	√
	*MQMALTUSR	USER DEF	-	-	-	-	√
PASS_ALL CONTEXT	*READ and *OBJOPR	*USE	√	√	-	-	-
	*MQMPASSALL	*USE	√	√	-	-	-
PASS_IDENTITY CONTEXT	*READ	USER DEF	√	-	-	-	-
	*MQMPASSID	USER DEF	√	-	-	-	-
SET_ALL CONTEXT	*ADD and *UPD	USER DEF	-	-	√	√	-
	*MQMSETALL	USER DEF	-	-	√	√	-
SET_IDENTITY CONTEXT	*UPD	USER DEF	-	-	√	-	-
	*MQMSETID	USER DEF	-	-	√	-	-

Table 4. Authorities for Context, AlternateUserID - AS/400 details

Opening MQSeries object with option MQOO_	MQSeries object (*OBJ) authority required - use GRTMQMAUT	MQSeries object authority shown - use DSPMQMAUT										
		Authority	Object					Data				
			Opr	Mgt	Exist	Alter	Ref	Read	Add	Upd	Del	Execute
ALTERNATE_USER AUTHORITY	*DLT	USER DEF	-	-	-	-	-	-	-	-	√	-
	*MQMALTUSR	USER DEF	-	-	-	-	-	-	-	-	√	-
PASS_ALL CONTEXT	*READ and *OBJOPR	*USE	√	-	-	-	-	√	-	-	-	-
	*MQMPASSALL	*USE	√	-	-	-	-	√	-	-	-	-
PASS_IDENTITY CONTEXT	*READ	USER DEF	-	-	-	-	-	√	-	-	-	-
	*MQMPASSID	USER DEF	-	-	-	-	-	√	-	-	-	-
SET_ALL CONTEXT	*ADD and *UPD	USER DEF	-	-	-	-	-	-	√	√	-	-
	*MQMSETALL	USER DEF	-	-	-	-	-	-	√	√	-	-
SET_IDENTITY CONTEXT	*UPD	USER DEF	-	-	-	-	-	-	-	√	-	-
	*MQMSETID	USER DEF	-	-	-	-	-	-	-	√	-	-

Security considerations for creating and changing channels

To use the CRTMQMCHL, CPYMQMCHL, or CHGMQMCHL commands you must have *ALLOBJ or be signed on to the system as QPGMR or QSYSOPR.

These commands allow the user to create or modify channel exits. Therefore, to prevent a potential security risk their usage is restricted.

If you are developing applications that modify channel exits, either get the Systems Administrator to create or modify the channel definitions, or ensure that a member of your team has the necessary authority.

MCAUSERID security considerations

When a user sends a message through a receiver channel their user ID has already been checked at the sending server end when the message was put to the remote queue. This means that all messages should be put to a queue through a receiver channel, and by using *NONE (or blanks) for the MCAUSERID field, the default user of QMQM is used.

The situation is slightly different when using clients with a server connection channel. In this case no security checking has been performed at the client end and, in order to get a message from a client application to an MQSeries queue, you must issue the appropriate authority to the queue manager and destination queue.

A client application contacts the queue manager through its defined server connection channel and the application adopts the authority of the MCAUSERID defined when the channel is created.

A newly created server connection channel, by default, has its MCAUSERID set to *PUBLIC. However this user identifier, by default, has been set to *EXCLUDE authority for all MQSeries objects, meaning that no access has been granted.

To resolve this situation you should change the MCAUSERID in the server connection channel to a specific user ID, which itself has been granted limited authority to the queue manager and destination queue.

Alternatively, you can give the user ID associated with MCAUSERID *PUBLIC authority to use the required MQSeries objects.

| **Note:** This option has the problem of permitting any access to the queue manager over that channel.

Default command authorities

The commands are shipped with the authorities described in "MQSeries for AS/400 commands having restricted usage" on page 273.

Note: These authorities can be changed for your installation.

With the exception of those commands listed other commands are shipped with public `*USE` authority.

Security considerations when designing an MQSeries for AS/400 application

When designing an MQM application you need to consider what access to MQSeries for AS/400 objects is required. You should only give access to those objects that are necessary, and with the minimum authority sufficient to perform the task.

For MQSeries for AS/400 objects, authority can be granted using the GRTMQMAUT command, and revoked using the RVKMQMAUT command. For MQSeries for AS/400 commands, use the standard AS/400 GRTOBJAUT and RVKOBJAUT commands.

The following list shows to what objects your application may require access:

- MQSeries for AS/400 message queue manager (`*MQM`)
- MQSeries for AS/400 catalog object (`*CTLG`)
- MQSeries for AS/400 administration object (`*ADM`)
- MQSeries for AS/400 queues (`*Q/*ALSQ/*LCLQ/*RMTQ/*MDL`)
- MQSeries for AS/400 processes (`*PRC`)
- MQSeries for AS/400 commands (`*CMD`)

For example, to give the user JOE the ability to display the message queue manager, using the DSPMQM command, issue the following:

```
GRTMQMAUT OBJ(QMGR) OBJTYPE(*MQM) USER(JOE) AUT(*READ)
```

Similarly, to give the user SUSAN ability to create and delete queues, using the CRTMQMQ and DLTMQMQ commands, issue the following:

```
GRTMQMAUT OBJ(QMGR) OBJTYPE(*CTLG) USER(SUSAN) AUT(*ADD *DLT)
```

Security considerations for users of the Administration utility

Authority to use the Administration utility can be controlled by granting and revoking authority to the STRMQMADM command, and the `QMQMADM` library, using the GRTOBJAUT and RVKOBJAUT commands.

When an operation is requested using the Administration utility, a command message is sent to the SYSTEM.ADMIN.COMMAND.QUEUE on the target system. Authorization checks are performed on the target system using the *UserIdentifier* from the message context of the command message.

One way of administering a remote system, is for users of the Administration utility to have a user ID on the remote system, being administered, that is the same as the user ID under which the Administration Utility is running on the local system.

For any operation, the user ID requires `*READ` authority to the message queue manager object on the target system.

In order to perform a Reset Queue Statistics operation, the user ID requires `*UPD` authority to the message queue manager object on the target system. In order to perform any other message queue manager, queue, or process operation, the user ID requires the MQSeries object authority on the target system that is required by the equivalent local CL command.

To perform any of the following operations, the user ID must have `*ALLOBJ` authority or be `QPGMR` or `QSYSOPR`:

- Ping channel
- Change channel
- Copy channel
- Create channel
- Delete channel
- Reset channel
- Start channel
- Stop channel
- Start channel initiator
- Start channel listener

AS/400 system authorities

In AS/400, the object authority that a particular MQSeries for AS/400 application has, cannot be based on the authority of a single user ID, because, in AS/400 a program has the ability to adopt the authority of the program that invokes it.

For example a user `'FRED'` may start an application called `'PAYROLL'` that uses another program called `'ACCOUNTS'` to update files on the system.

FRED and PAYROLL have no authority to update any of the account records and ACCOUNTS has no authority to access the personnel files.

The program ACCOUNTS adopts the object authority of the programs earlier in the invocation stack, so ACCOUNTS can access the personnel files and update the account files.

The same is true of an MQSeries for AS/400 application. The authority for a program to access a particular MQSeries for AS/400 object is dependent upon the access permissions of the issuing program.

In the above example, if an MQOPEN call is issued in ACCOUNTS, the object authority is FRED + PAYROLL + ACCOUNTS. It is not, therefore, possible in AS/400 to pass a single user ID as a means of querying access authority.

Chapter 4. Operating MQSeries for AS/400 using CL commands

This chapter gives an overview of working with MQSeries for AS/400 from the AS/400 command line, together with some suggested operations.

Whenever you start any job, the system creates a QTEMP library. When you connect the job to the queue manager you may get various objects of type *USRSPC created. These objects are detailed in Appendix C, "Specific MQSeries for AS/400 considerations" on page 273.

When you have created the local queue manager, you are strongly recommended to use the CCTMQM command when you are working with MQSeries for AS/400 objects, apart from channels.

The use of this command can lead to a significant improvement in performance, especially when you are batching jobs. If you use the CCTMQM command, you should use the DSCMQM command at the end of the group of commands.

MQSeries applications

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

- Recovery purposes
- Maintenance
- Rolling out MQSeries applications

You can do this by either:

- Creating CL programs to generate your MQSeries definitions for the AS/400, or
- Creating MQSC text files to generate your MQSeries definitions using the cross-platform MQSeries command language.

Format of MQSeries for AS/400 commands

The general form of a command name is xxxMQMooo, where the:

- Prefix xxx is the verb
- Suffix ooo is the object

For example, CHGMQMPRC is the command that changes an MQSeries for AS/400 process. Table 5 on page 32 shows some of the commands that are available where the:

| - Prefix is represented by the Option column
| - Suffix is represented by the other columns, if applicable

© Copyright IBM Corp. 1994, 1998

Table 5. Command Matrix

Option	Local Queue	Model Queue	Remote Queue	Alias Queue	Process	Channel
Create (CRT)	Q	Q	Q	Q	PRC	CHL
Copy (CPY)	Q	Q	Q	Q	PRC	CHL
Change (CHG)	Q	Q	Q	Q	PRC	CHL
Display (DSP)	Q	Q	Q	Q	PRC	CHL
Delete (DLT)	Q	Q	Q	Q	PRC	CHL
Work with (WRK)	Q	Q	Q	Q	PRC	CHL
Clear (CLR)	Q	N/A	N/A	N/A	N/A	N/A
Start (STR)	N/A	N/A	N/A	N/A	CHL	
End (END)	N/A	N/A	N/A	N/A	N/A	CHL
Reset (RST)	N/A	N/A	N/A	N/A	N/A	CHL
Resolve (RSV)	N/A	N/A	N/A	N/A	N/A	CHL

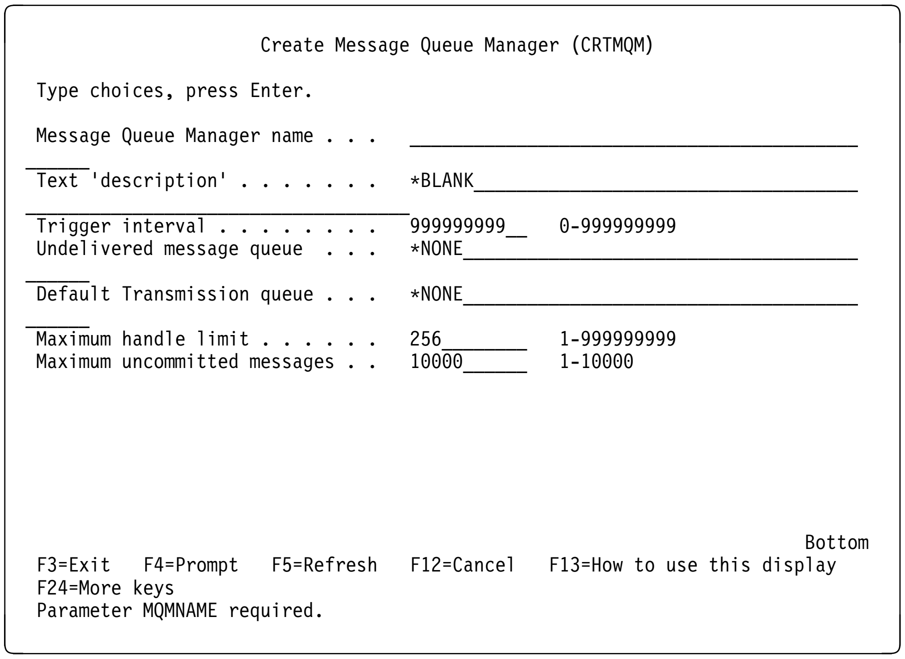
Verifying the installation

When you have installed MQSeries for AS/400 on your system, you should:

1. Create a queue manager. To do this you:

   a. Issue the CRTMQM command from the command line and press the `Enter` key. The screen shown in Figure 2 appears if the product has installed correctly, displaying the message `Parameter MQMNAME required` at the bottom of the screen.

      For details of the CRTMQM command, see “CRTMQM (Create Message Queue Manager) Command” on page 155.



```
                   Create Message Queue Manager (CRTMQM)

 Type choices, press Enter.

 Message Queue Manager name . . .   __________________________________________

 Text 'description' . . . . . . .   *BLANK____________________________________

 Trigger interval . . . . . . . .   999999999__   0-999999999
 Undelivered message queue  . . .   *NONE_____________________________________

 Default Transmission queue . . .   *NONE_____________________________________

 Maximum handle limit . . . . . .   256_______    1-999999999
 Maximum uncommitted messages . .   10000_____    1-10000






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

Figure 2. CRTMQM panel

If the screen does not appear, ensure that the tape loaded correctly see "Verifying the product installation" on page 16. If the tape loaded correctly, see "Deleting MQSeries for AS/400 V4R2M1" on page 18 to delete MQSeries for AS/400, and then install the MQSeries for AS/400 `*BASE` tape again.

b. Type in a Message Queue Manager name and press the `Enter` key.

   The name of the queue manager can be a maximum of 48 characters in length. Valid characters are detailed in "Names" on page 102.

2. Start the queue manager from the command line by issuing the command `STRMQM MQMNAME(nnnnn)`, where `nnnnn` is the name of the queue manager that you have just created.

   After a short period you receive the message `Message Queue Manager started`.

3. Set up the default objects by issuing the command `CALL QMQM/AMQSDEF4`.

   For a list of the default objects supplied, see "System defaults" on page 103.

Notes:

1. The source for AMQSDEF4 is shown in "AMQSDEF4" on page 241. The `AMQSDEF4` program is supplied in a compiled form with the base product, and as source code on the sample tape, in library `QMQMSAMP`, as object `QCLSRC`, member `AMQSDEF4`.

2. If you want to change any of the attribute values, or names, you must ensure that you have loaded the samples tape.

   To change the default definitions, you edit and compile member AMQSDEF4 and run the CL program created, using any standard method. The Work with Objects using PDM (WRKOBJPDM) command is one way to edit, compile and run this member.

3. As supplied, any `SYSTEM.DEFAULT.QUEUE` has `PUBLIC(*READ)` authority.

If these operations are successful you should initialize the administration utility, if you have installed this option, by:

1. Running the sample program AMQSADM4, by issuing the command CALL QMQM/AMQSADM4 USERID, where USERID is the user ID of the administrator running the application, for example, QSECOFR.

   The user ID must be a valid, uppercase, AS/400 user ID.
2. Starting the Administration utility by issuing the STRMQMADM command with STRMQM(*YES) specified. See "Verifying the Administration utility" on page 46 for more details on how to verify the Administration utility.

The source of AMQSADM4 is shown in "AMQSADM4" on page 253. The AMQSADM4 program is supplied in a compiled form with the base product, and as source code on the Administration utility tape.

If these operations are not successful, you should delete the product and install it again.

MQSeries for AS/400 libraries

When installation has completed, the system has created various libraries, which you do not have to add to your library list. All these libraries start with the character string QMQM, and include:

QMQM	– base product
QMQMDATA	– data library
QMQMPROC	– data library
QMQMSAMP	– samples (optional)
QMQMADM	– Administration utility (optional)

Customizing MQSeries for AS/400 V4R2M1

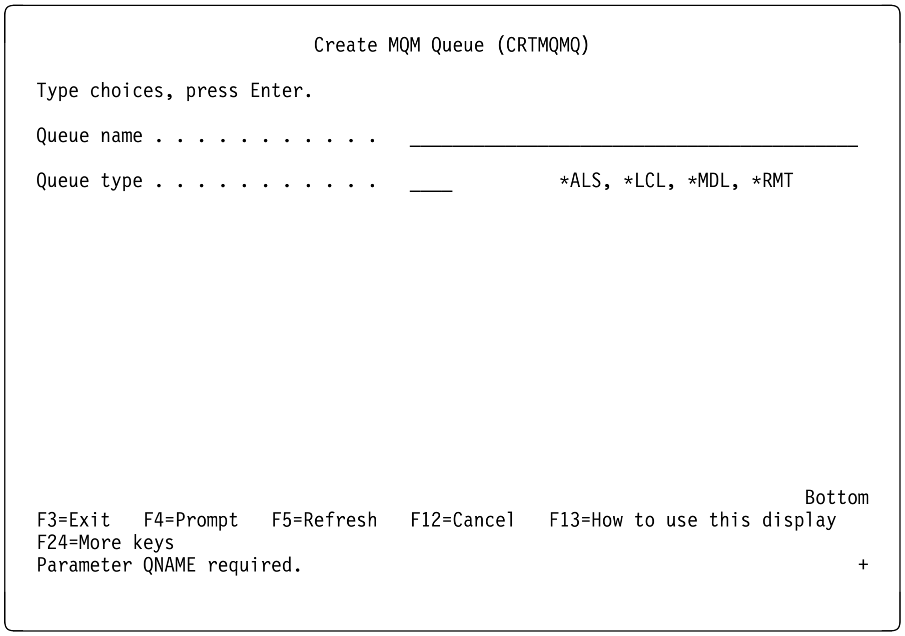
MQSeries for AS/400 requires some customization after installation in order to meet the individual and special requirements of your system, and to use your system resources in the most effective way.

You need to:

- Define queues, including an undelivered message queue
- Set up remote links – see the *MQSeries Intercommunication* book for information on how to do this.

You create an undelivered message queue by:

1. Issuing the CRTMQMQ command from the command line and pressing the PF4 key, which displays the panel shown in Figure 3.



```
                       Create MQM Queue (CRTMQMQ)

 Type choices, press Enter.

 Queue name . . . . . . . . . .   ________________________________________

 Queue type . . . . . . . . . .   ____           *ALS, *LCL, *MDL, *RMT













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

Figure 3. CRTMQMQ panel

2. Entering the name of the queue, that you are going to use as the undelivered-message queue, in the `QNAME` field, specifying a maximum of 48 characters for the name.

   Note: If you used `AMQSDEF4` to create the default objects, the name of the undelivered-message queue created by the program is `SYSTEM.DEAD.LETTER.QUEUE`

3. Specifying `*LCL` in the `QTYPE` field.
4. Pressing the `Enter` key which takes you to the optional parameter panels.
5. Setting the *Maxdepth* and *Maxmsglen* attributes to the maximum value possible, that is, 640 000 and 4 194 304.

   You do not need to change any of the other default values.

6. Pressing the `Enter` key to create the queue.
7. Issuing the CHGMQM command from the command line and changing the name of the undelivered message queue field from `*NONE` to the name of the queue that you have just created.

For details of the CRTMQMQ command, see "CRTMQMQ (Create MQM Queue) Command" on page 168.

Note: You should define and agree a naming convention within your enterprise for queue names, because this will avoid problems where an application attempts to address a nonexistent queue.

In this situation the message is not discarded, but placed on an undelivered-message (dead-letter) queue. It is your responsibility to decide on the disposition of messages appearing on the undelivered-message queue. You can:

- Redirect these messages to another queue
- Delete the messages

- Return the messages to their original sender where possible
- Ignore the message

In the last situation, a danger exists that the undelivered-message queue will become full. If this happens, no further messages can be placed on the queue until space becomes available and the messages will be lost. Therefore, you should regularly inspect the undelivered-message queue and dispose of any messages on it.

Starting the local queue manager

The local queue manager must first be created by issuing the CRTMQM command. It is then started by issuing the STRMQM command from an AS/400 command line. It can be stopped by issuing ENDMQM from the AS/400 command line.

The queue manager can be controlled issuing commands from an AS/400 command line. These commands are described later in this chapter . However, if you prefer to use a panel interface to the queue manager, an administration utility is available with MQSeries for AS/400. This facility is detailed in Chapter 5, “The MQSeries for AS/400 Administration utility” on page 43.

The Administration utility is started by issuing the command STRMQMADM from a command line. When started, the MQSeries Administration main menu is displayed.

From this menu you may add queue manager groups, which then become persistent until you remove them. Persistent queue manager groups reappear whenever the MQSeries Administration Utility is started, until they are deleted.

You can perform most commands and operations on both local and remote queue managers. However, for remote queue managers, the following restrictions apply:

- Starting and ending the queue manager
- Starting, ending, and displaying the command server
- Resolving channels
- Granting, displaying, and revoking authorities
- Displaying channel status
- Working with messages

Queue manager groups are used to group related queue managers. This relationship is installation dependent. For example, queue managers can be grouped geographically, by department, or by any other grouping of your enterprise’s choice.

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 MQSeries for AS/400) exist to enable such operations.

The local queue administrator cannot stop a remote queue manager.

Creating MQSeries objects

The following tasks suggest various ways in which you can use MQSeries for AS/400, from the command line. Where appropriate, references are also given to the appropriate sections of the chapter on the MQSeries Administration utility.

There are three online methods to create MQSeries objects, which are:

1. Using a Create command

CRTMQMCHL	Create MQM Channel
CRTMQMPRC	Create MQM Process
CRTMQMQ	Create MQM Queue

2. Using the appropriate Work with MQM object command

WRKMQMCHL	Work with MQM Channels
WRKMQMPRC	Work with MQM Processes
WRKMQMQ	Work with MQM Queues

3. Using the MQSeries administration utility. Details of using this utility are given in Chapter 5, “The MQSeries for AS/400 Administration utility” on page 43.

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, you can:

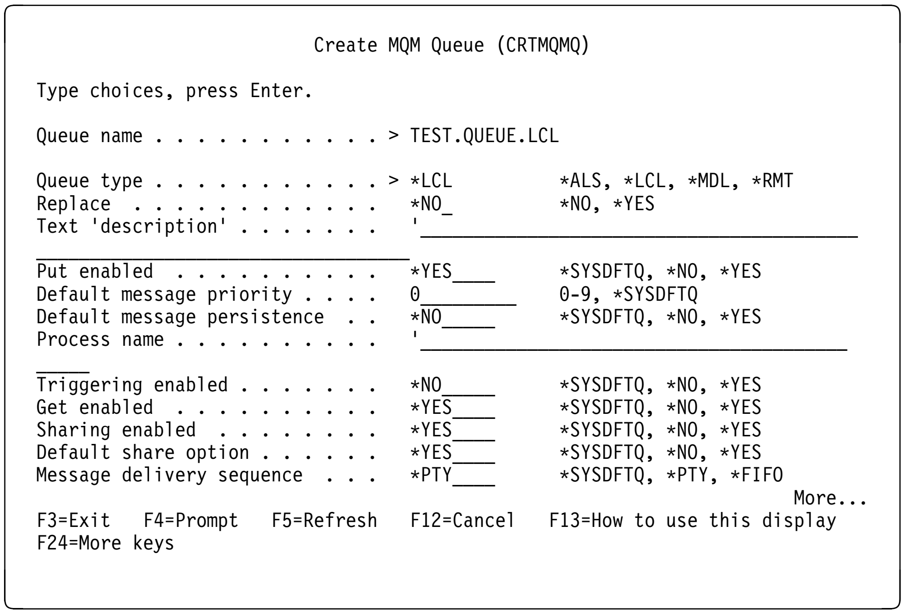
1. Use the Create MQM Queue (CRTMQMQ) command
2. Use the Work with MQM Queues (WRKMQMQ) command

Creating a local queue using the CRTMQMQ command

1. Type CRTMQMQ on the command line and press the `PF4` key. This displays Figure 3 on page 35.
2. On the Create MQM Queue panel, type the name of the queue 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, and press the `Enter` key. Further settings for a local queue will be displayed, see Figure 4, with the fields containing the default values. You may overtype any of these values with a new value.

   Scroll forward to see further fields.
4. When you have made any changes to the values, press the `Enter` key to create the queue.



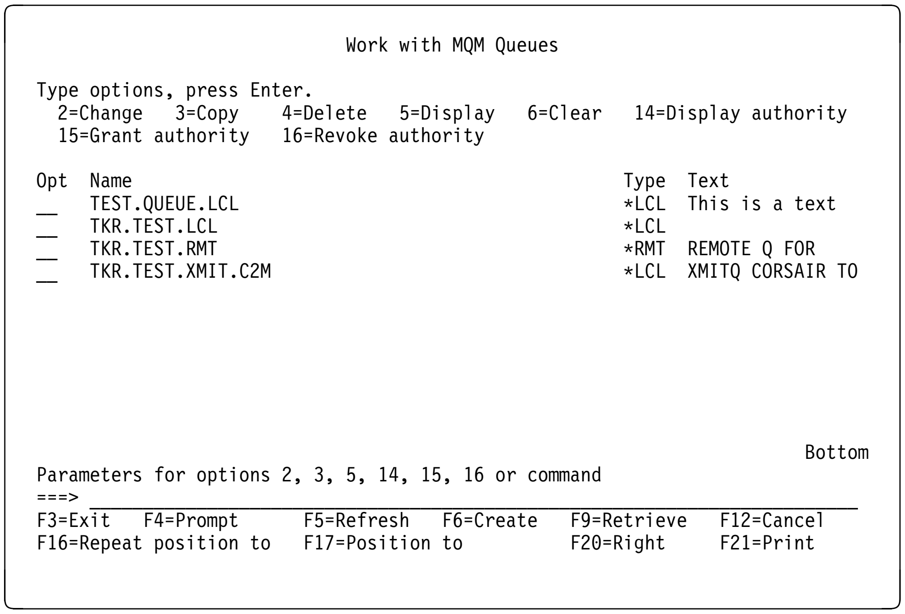
```
                         Create MQM Queue (CRTMQMQ)

Type choices, press Enter.

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

Queue type . . . . . . . . . . . > *LCL         *ALS, *LCL, *MDL, *RMT
Replace  . . . . . . . . . . . .   *NO_         *NO, *YES
Text 'description' . . . . . . .   '_________________________________________
___________________________________
Put enabled  . . . . . . . . . .   *YES____      *SYSDFTQ, *NO, *YES
Default message priority . . . .   0_________    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
Default share option . . . . . .   *YES____      *SYSDFTQ, *NO, *YES
Message delivery sequence  . . .   *PTY____      *SYSDFTQ, *PTY, *FIFO
                                                                       More...
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 4. Create MQM Queue options panel

Creating a local queue using the WRKMQMQ command

1. Type WRKMQMQ on the command line.
2. 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.
3. Press the Enter key and Figure 5 is displayed.



```
                            Work with MQM Queues

Type options, press Enter.
  2=Change   3=Copy    4=Delete   5=Display   6=Clear   14=Display authority
  15=Grant authority   16=Revoke authority

Opt  Name                                            Type  Text
__   TEST.QUEUE.LCL                                  *LCL  This is a text
__   TKR.TEST.LCL                                    *LCL
__   TKR.TEST.RMT                                    *RMT  REMOTE Q FOR
__   TKR.TEST.XMIT.C2M                               *LCL  XMITQ CORSAIR TO









                                                                        Bottom
Parameters for options 2, 3, 5, 14, 15, 16 or command
===> ____________________________________________________________________
F3=Exit   F4=Prompt      F5=Refresh   F6=Create   F9=Retrieve   F12=Cancel
F16=Repeat position to   F17=Position to          F20=Right     F21=Print
```

Figure 5. Work with MQM Queues panel

4. Press F6 to create a new queue, which takes you to the CRTMQMQ panel. See "Creating a local queue using the CRTMQMQ command" on page 37 for instructions on how to create the queue.

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

Examples of creating a remote queue

You use the panel shown in Figure 3 on page 35, 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.
3. Option `1=Create` on the MQSeries Work with Objects panel of the MQSeries Administration utility. See "Work with objects" on page 51 for details on how to use this option.

The use of remote queues is described in detail in the *MQSeries Intercommunication* book.

This section describes how to define a remote queue for each of the three uses.

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, you:

1. Display the `Create MQM Queue` panel.
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 at the remote location in the `Remote queue` field.
5. Type the name of the queue manager at the remote location in the `Remote Message Queue Manager` field.

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 the *MQSeries Intercommunication* book for further information.

To define a remote queue as a queue manager alias, you:

1. Display the `Create MQM Queue` panel.
2. Type the queue name in the `Queue name` field.
3. Type `*RMT` in the `Queue type` field.
4. Type the name of the queue manager at the remote location in the `Remote Message Queue Manager` field.

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

An application may 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, you define a remote queue with the same name as the reply-to queue.

See the *MQSeries Intercommunication* book for further information.

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

1. Display the `Create MQM Queue` panel.
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 queue name in the `Queue name` field.

   This is the name of the queue to which you want the reply-to messages sent.
5. 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.

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.

Applications can put messages directly on a message queue, or they can be put there indirectly, for example, through a remote queue definition.

To create a transmission queue, you:

1. Display the `Create MQM Queue` panel.
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.

For more information about transmission queues and message channels, see the *MQSeries Intercommunication* book.

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 *MQSeries Application Programming Guide*.

Creating an alias queue

You 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, you:

1. Display the `Create MQM Queue` panel.
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 that you want the queue name to resolve to in the `Target queue` field.

Creating a model queue

You 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.)

Chapter 5. The MQSeries for AS/400 Administration utility

This chapter describes how to define and manage MQSeries for AS/400 resources using the MQSeries for AS/400 Administration utility. Some suggested tasks, using the utility, are described in Appendix B, “Administration utility – example tasks” on page 269. These tasks are:

- “Creating a local queue” on page 269
- “Setting up a link between two queue managers” on page 269
- “Checking queue statistics” on page 272

The MQSeries for AS/400 Administration utility is used to manage the queue managers for which you have authority interactively. The queue manager that you want to manage must have been created.

Note: You cannot use the Administration utility to create or delete a message queue manager.

You can use the MQSeries for AS/400 Administration utility to:

- Start the local queue manager, and the local queue manager’s command server.
- Create, copy, and delete local and remote MQSeries for AS/400 objects.
- Display and change the attributes of a queue manager, either local or remote, and its associated objects.
- Grant and revoke security access to local queue manager objects.
- View the contents of a queue and individual messages, on the local queue manager only.

Setting up the Administration utility

If you did not start the Administration utility when you installed MQSeries for AS/400 (see “Verifying the installation” on page 32) you need to create the following objects by issuing the command `CALL QMQM/AMQSADM4 userid`, where `userid` is the user ID of the person who is going to administer the Administration utility:

- `SYSTEM.ADMIN.MQMLIST.`*userid*, which records the queue managers that your ID is administering.
- `SYSTEM.ADMIN.REPLYQ.`*userid*, which is the reply to queue for remote commands and cache request processing.
- `SYSTEM.ADMIN.RESPQ.`*userid*, which tracks remote commands and responses.
- `SYSTEM.ADMIN.EXCEPTION.QUEUE`, where messages are placed that have caused internal processing errors, or where unrecognized messages are placed.
- `SYSTEM.ADMIN.AMQMDATA.QUEUE`, the queue for requests to the local data-store server.

Working with the Administration utility

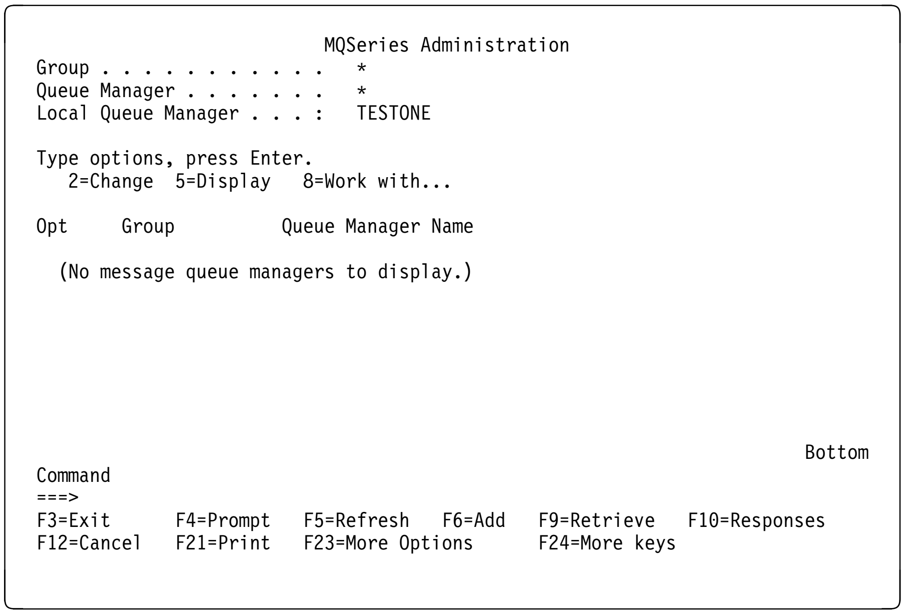
You start the MQSeries for AS/400 Administration utility by entering STRMQMADM on the command line.

Note: See "STRMQMADM (Start MQM Administrator) Command" on page 219 for a detailed explanation of the STRMQMADM command.

To start the queue manager, you *must* specify *STRMQM*(*YES) and to start the command server you *must* specify *STRCSVR*(*YES).

When the utility starts you are presented with the MQSeries Administration main menu (see Figure 6), displaying the message shown.

The main menu presents you with various options. Depending on which option you select, you are guided through a series of menus and options until the Administration utility has enough information to call the appropriate MQSeries for AS/400 Command Language (CL) command, or send a command to the command server of the selected queue manager.



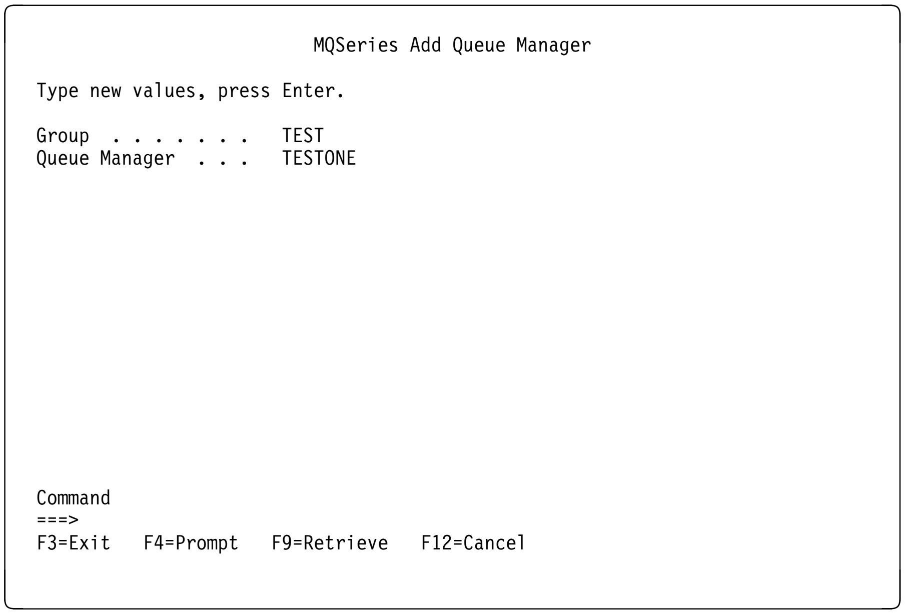
```
                           MQSeries Administration
Group . . . . . . . . . . .   *
Queue Manager . . . . . . .   *
Local Queue Manager . . . :   TESTONE

Type options, press Enter.
   2=Change  5=Display   8=Work with...

Opt     Group          Queue Manager Name

  (No message queue managers to display.)






                                                                      Bottom
Command
===>
F3=Exit      F4=Prompt   F5=Refresh   F6=Add   F9=Retrieve   F10=Responses
F12=Cancel   F21=Print   F23=More Options      F24=More keys
```

Figure 6. Initial MQSeries Administration main menu

You must first add a queue manager, which is usually the local queue manager whose name is displayed on this panel.

Pressing Function Key 6 (F6), takes you to the panel shown in Figure 7 on page 45, on which you enter the following information:

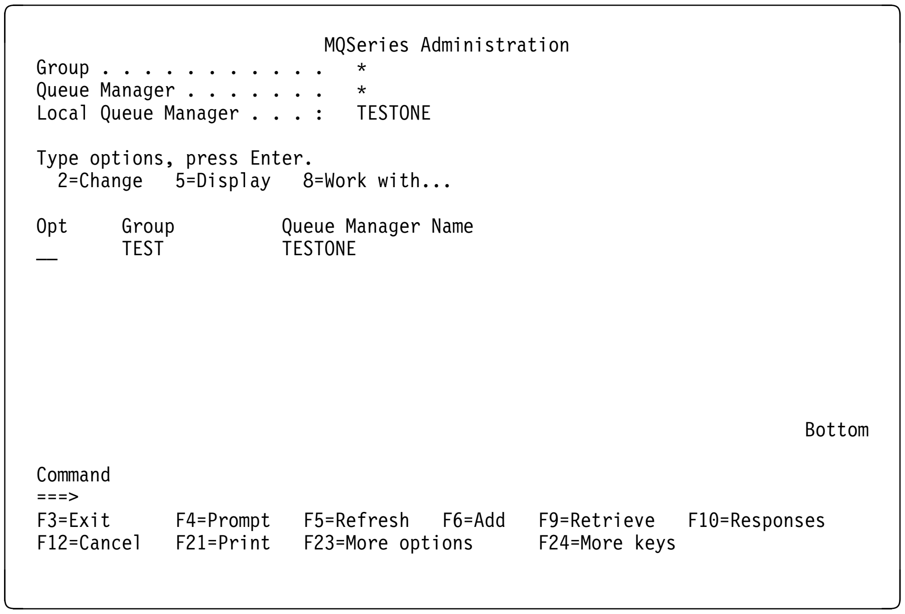
- Group name
- Queue Manager name



```
                         MQSeries Add Queue Manager

Type new values, press Enter.

Group  . . . . . . .   TEST
Queue Manager  . . .   TESTONE
















Command
===>
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
```

Figure 7. MQSeries Add Queue Manager panel

Typing the text shown and pressing the `Enter` key changes the main menu to display the menu shown in Figure 8.



```
                          MQSeries Administration
Group . . . . . . . . . . .   *
Queue Manager . . . . . . .   *
Local Queue Manager . . . :   TESTONE

Type options, press Enter.
  2=Change   5=Display   8=Work with...

Opt     Group          Queue Manager Name
__      TEST           TESTONE








                                                                       Bottom

Command
===>
F3=Exit      F4=Prompt   F5=Refresh   F6=Add   F9=Retrieve   F10=Responses
F12=Cancel   F21=Print   F23=More options      F24=More keys
```

Figure 8. MQSeries Administration main menu – queue manager displayed

Verifying the Administration utility

You should verify that the Administration utility is working correctly with a local queue by:

1. Adding a local queue manager.
2. Enter `30`, in the Option column, against the queue manager. This sends a ping to the command server of that queue manager.
3. Press Function Key 10 (`F10`) to display the response.
4. You should obtain a line of text similar to the first line of the display shown in Figure 18 on page 59.
5. If the `Avl` field is `N`, check that the command server is running by working with the DSPMQMCSVR command, and press Function Key 5 (`F5=Refresh`).

Administering remote queue managers

You can use the Administration utility to administer remote queue managers by adding a message queue manager to a remote system.

To do this, you must have a user ID on that system that is the same as the user ID on the local system, and ensure that the command server is started on that system.

Note: If the remote queue manager is on a Windows NT or UNIX system, the user ID QMQM must exist on that remote system. The user ID QMQM always exists on the system where MQSeries for AS/400 is installed.

For the Administration utility to communicate with the remote message queue manager, you must create a link to:

1. Send messages to the remote queue manager
2. Receive messages from the remote queue manager

You are recommended to use the default transmission queue for each local message local queue manager.

You do not need to set up a remote queue for each of the queues that the message queue manager uses on the remote system. The Administration utility uses one queue on each message queue manager to send messages to the other message queue manager. These are transmission queues and they ***must*** have the same name as the name of the message queue manager to which the messages are being sent.

An example of setting up links between two queue managers using TCP/IP is provided in “Setting up a link between two queue managers” on page 269.

For information on communicating between message queue managers, see the *MQSeries Intercommunication* book.

Notes:

1. To add further queue managers, press Function Key 6 (`F6`) and complete the details for each queue manager that you want to add.

   To move a queue manager from one group to another, you:

   a. Select option `21=Remove` from the MQSeries Administration main menu.

   b. Press the `Enter` key on the removal confirmation panel that is displayed.
   c. Press function key six (`PF6=Add`) on the MQSeries Administration main menu, to add a queue manager to the list.
   d. Type the new group name and the queue-manager name on the MQSeries Add Queue Manager panel and press the `Enter` key.
2. The main panel presents a list of queue managers, for which you are responsible.
3. The queue-manager name is case sensitive. If you use lower-case characters you ***must*** enclose the name in apostrophes.
4. A group is created when you create the first queue manager in that group and deleted when the last queue manager in that group is deleted.
5. All responses from the Administration utility are sent to the queue, SYSTEM.ADMIN.RESPQ.`userid`.
6. You can enhance the performance of the Administration utility further by altering the dispatching priority of the administration job.

   As the dispatching priority is an attribute of the job, rather than the application itself, adjustment can improve the performance of the utility. However, adjusting the priority of the job affects the performance of the overall system; therefore, you need to experiment and decide whether the adjustment is useful and acceptable.

Options for Administration utility main menu

The main menu panel (see Figure 6 on page 44) allows you to select the following options:

2=Change
Enables you to change the selected queue manager attributes.

5=Display
Enables you to display information about the selected queue manager.

8=Work with...
Takes you to Figure 12 on page 51. In the example illustrated, you selected `TESTONE`.

10=Display CmdServer
Enables you to display information about the local queue-manager command server.

11=Start CmdServer
Enables you to start the local queue-manager command server.

14=Stop CmdServer
Enables you to stop the local queue-manager command server.

21=Remove
Enables you to remove the selected local queue-manager from the list.

30=Ping Queue Mgr
Sends a ping to the command server of the selected queue manager. This option is useful for testing communications to a remote queue manager.

Function keys for Administration utility main menu

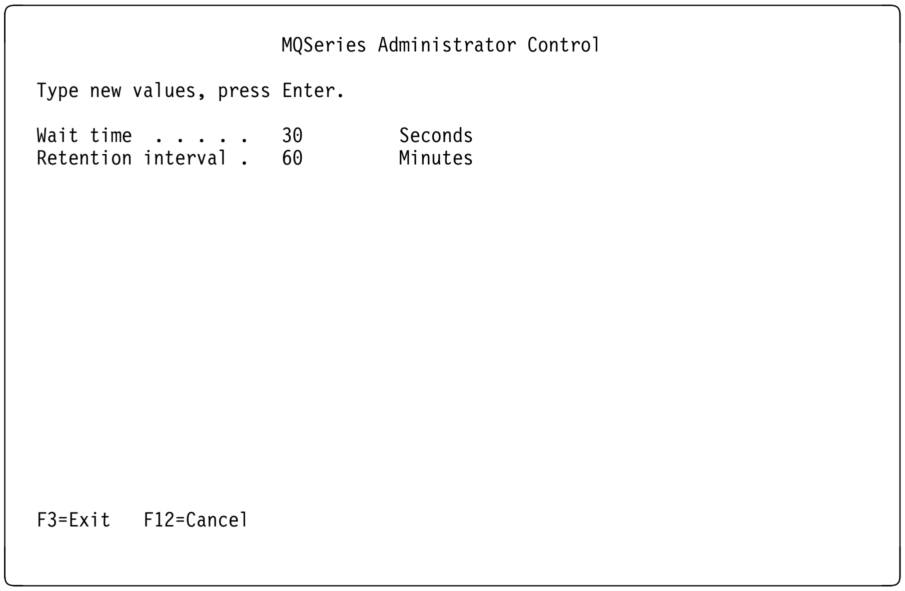
F6=Add
Takes you to the MQSeries Add Queue Manager panel shown in Figure 7 on page 45.

F10=Responses
Takes you to the Work with Responses panel shown in Figure 18 on page 59.

F21=Print
Sends the list currently selected to QSYSPRT.

F22=Options
Takes you to the MQSeries Administrator Control panel shown in Figure 9.

Administrator control panel



```
                      MQSeries Administrator Control

 Type new values, press Enter.

 Wait time  . . . . .   30         Seconds
 Retention interval .   60         Minutes

















 F3=Exit   F12=Cancel
```

Figure 9. MQSeries Administrator Control panel

This panel displays values that control the operation of the Administration utility. You can change either, or both, of the values, which are:

Wait time
This option allows you to set the maximum time, in seconds, for a response to be returned.

The value is dependent upon the network to which you are connected, and how long you want to wait.

The valid range is 1 through 3600 seconds, and the default setting is 30 seconds.

As a guide, you are recommended to set this field to a period of 900 seconds, for a large network.

For example, if you set the time to 30 seconds and, at the end of this period, the data is not available, you have the option of either waiting for a further period of 30 seconds, or continuing with another task and returning subsequently to request the data again.

Retention interval
This option allows you to set the interval, in minutes, after which any object in the Administration data store is considered to be out of date.

The valid range is 0 through 1440 minutes (one day), and the default setting is 60 minutes.

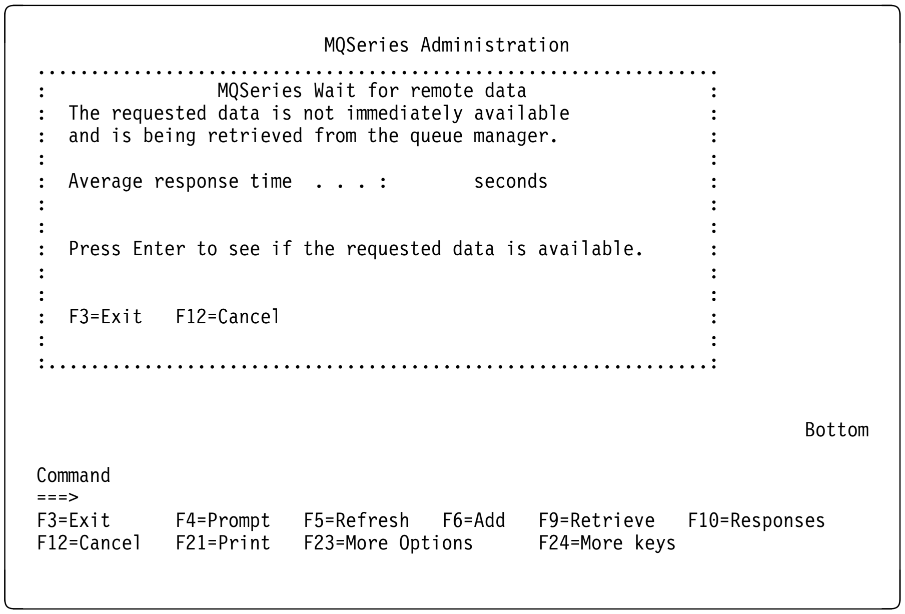
Setting this value to zero allows you always to request data from the remote queue manager.

Details of queue managers and their associated objects are kept in a local data store, in order to improve the time taken to display information, and to reduce network traffic.

If a request is made to display an object and the object data is considered to be out of date, or is not present, a new copy is requested from the queue manager. There is one exception to this, however, and this occurs when the object data has not been viewed. This is to avoid not being able to display an object, if the Retention Interval is set to a lower value than the time taken to view the object after it has arrived in the data store.

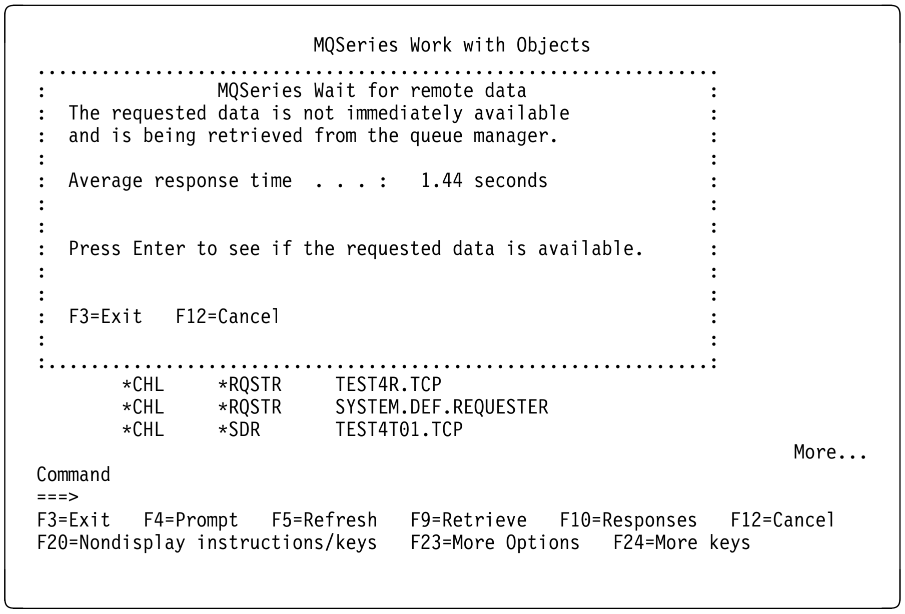
If you select option **5=Display** from either the MQSeries Administration main menu, or the MQSeries Work with Objects menu, you obtain one of the following secondary windows, when data is not available locally.

If there is no response time available, the panel shown in Figure 10 is displayed; if the response time is available, the panel shown in Figure 11 on page 50 is displayed.



```
                             MQSeries Administration
 ................................................................
 :                MQSeries Wait for remote data                  :
 :  The requested data is not immediately available              :
 :  and is being retrieved from the queue manager.               :
 :                                                               :
 :  Average response time  . . . :        seconds                :
 :                                                               :
 :                                                               :
 :  Press Enter to see if the requested data is available.       :
 :                                                               :
 :                                                               :
 :  F3=Exit   F12=Cancel                                         :
 :                                                               :
 :...............................................................:


                                                                       Bottom

 Command
 ===>
 F3=Exit      F4=Prompt   F5=Refresh   F6=Add   F9=Retrieve   F10=Responses
 F12=Cancel   F21=Print   F23=More Options      F24=More keys
```

Figure 10. Response time unavailable



```
                          MQSeries Work with Objects
................................................................
:                MQSeries Wait for remote data                 :
:  The requested data is not immediately available             :
:  and is being retrieved from the queue manager.              :
:                                                              :
:  Average response time  . . . :   1.44 seconds              :
:                                                              :
:                                                              :
:  Press Enter to see if the requested data is available.      :
:                                                              :
:                                                              :
:  F3=Exit   F12=Cancel                                        :
:                                                              :
:..............................................................:
        *CHL     *RQSTR     TEST4R.TCP
        *CHL     *RQSTR     SYSTEM.DEF.REQUESTER
        *CHL     *SDR       TEST4T01.TCP
                                                                      More...
Command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Responses   F12=Cancel
F20=Nondisplay instructions/keys   F23=More Options   F24=More keys
```

Figure 11. Response time available

The average response time is the average time taken for data requests to be received from the selected queue manager.

Note: The average response time is calculated from a maximum of the last eight responses received.

If the response time is blank, no responses have been received. This is because:

- No requests have been made, or
- Requests have been made and no response has yet been received. This condition may be due to:
  - The command server on the selected queue manager not being started.
  - Delays in the network.

Pressing the `Enter` key checks to see if the response data has become available.

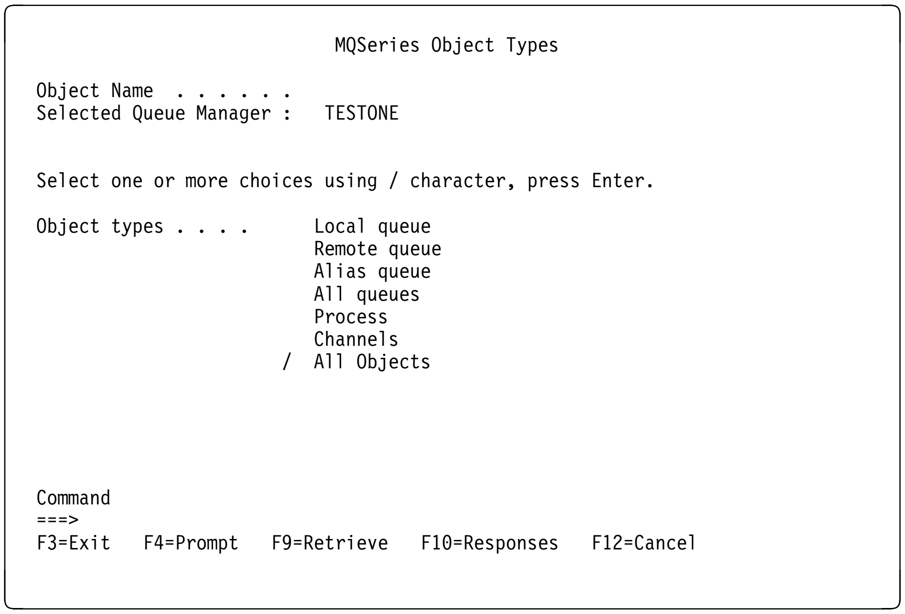
The request can be canceled by pressing Function Key 12 (`F12`).

Work with queue-manager object types

This menu, selected by choosing option **8=Work with...** from the MQSeries Administration main menu, displays a list of the types of MQSeries for AS/400 objects, for the selected queue manager, on which you:

- Enter an object name, which can be a generic name
- Select the type, or types, of object that you want to work with

In the example illustrated you are selecting `All Objects`.



```
                            MQSeries Object Types

Object Name  . . . . . .
Selected Queue Manager :   TESTONE


Select one or more choices using / character, press Enter.

Object types . . . .      Local queue
                          Remote queue
                          Alias queue
                          All queues
                          Process
                          Channels
                       /  All Objects





Command
===>
F3=Exit   F4=Prompt   F9=Retrieve   F10=Responses   F12=Cancel
```

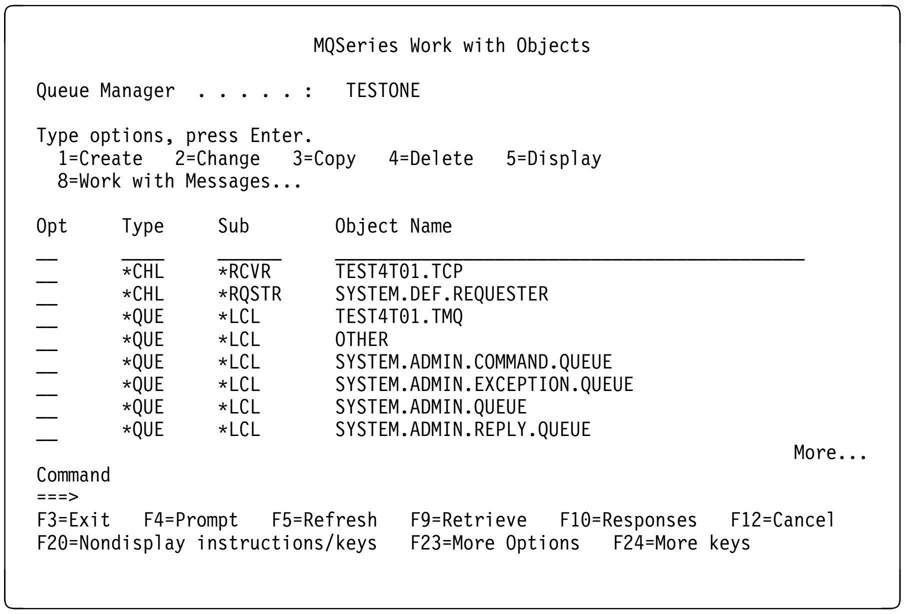
Figure 12. MQSeries Object Types panel

After pressing the `Enter` key, the information is requested from the selected queue manager and the MQSeries Work with Objects panel is displayed.

Work with objects

This panel lists the objects that match the selected combination of name and type. Here you identify, as an option number, an activity, and an object name.

The base menu is shown in Figure 13.



```
                          MQSeries Work with Objects

Queue Manager  . . . . . :   TESTONE

Type options, press Enter.
  1=Create   2=Change   3=Copy   4=Delete   5=Display
  8=Work with Messages...

Opt     Type     Sub        Object Name
__      ____     ______     ____________________________________________
__      *CHL     *RCVR      TEST4T01.TCP
__      *CHL     *RQSTR     SYSTEM.DEF.REQUESTER
__      *QUE     *LCL       TEST4T01.TMQ
__      *QUE     *LCL       OTHER
__      *QUE     *LCL       SYSTEM.ADMIN.COMMAND.QUEUE
__      *QUE     *LCL       SYSTEM.ADMIN.EXCEPTION.QUEUE
__      *QUE     *LCL       SYSTEM.ADMIN.QUEUE
__      *QUE     *LCL       SYSTEM.ADMIN.REPLY.QUEUE
                                                                  More...
Command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Responses   F12=Cancel
F20=Nondisplay instructions/keys   F23=More Options   F24=More keys
```

Figure 13. MQSeries Work with Objects - base options

Options for Work with Objects

This panel allows you to select the following options:

1=Create

To create an MQSeries object you need to specify the object type and, for channels and queues, a subtype.

Channels (*CHL) require a subtype, which can be one of:

- `*RCVR` – receiver
- `*SDR` – sender
- `*RQSTR` – requester
- `*SVR` – server
- `*SVRCN` – server connection

Pressing the Enter key, or Function Key 4 (`PF4`), displays the create channel command. This allows you to complete the definition.

Queues (*QUE) require a subtype, which can be one of:

- `*ALS` – alias
- `*LCL` – local
- `*MDL` – model
- `*RMT` – remote

Pressing the Enter key, or Function Key 4 (`PF4`), displays the create queue command. This allows you to complete the definition.

A process (*PRC) does not have a subtype. Pressing the Enter key, or Function Key four (`PF4`), displays the create process command. This allows you to complete the definition.

2=Change

Can be entered against a channel, queue, or process. The relevant change command is displayed, allowing you to change the definition of the object.

3=Copy

Can be entered against a channel, queue, or process. The relevant copy command is displayed, allowing you to create a copy of the object.

4=Delete

Can be entered against a channel, queue, or process. In order for the delete option to be successful, the object in question must not be in use.

5=Display

Can be entered against a channel, queue, or process to display the object definition.

8=Work with Messages

Displays the panel shown in Figure 15 on page 55.

Note: This option can be entered only if you are working with a queue on the local queue manager. If you are working with a remote queue manager, an asterisk character (*) appears on the display to show that the option is unavailable. Further details are given in “Options for Work with Messages” on page 55.

12=Reset Queue

Can be entered only against a queue and sends a Reset Queue Statistics command to the selected queue. The results can be viewed by pressing Function Key 10 (`PF10`).

13=Clear Queue
Can be entered only against a queue and enables you to clear a specified queue.

14=Display Authority
Can be entered only against a queue or process. This option enables you to see, for a specified local object, the list of authorized users for that object, and the authorities for the object.

Note: This option is available only if you are working with the local queue manager. If you are working with a remote queue manager, an asterisk character appears on the display to show that the option is unavailable.

15=Grant Authority
Can be entered only against a queue or process. This option enables you to grant specific authority, for the named local object, to another user, or group of users.

Note: This option is available only if you are working with the local queue manager. If you are working with a remote queue manager, an asterisk character appears on the display to show that the option is unavailable.

16=Revoke Authority
Can be entered only against a queue or process. This option enables you to remove specific authorities for named local objects, from one or more users named in the command, or to remove the authority of an authorization list for the named objects.

Note: This option is available only if you are working with the local queue manager. If you are working with a remote queue manager, an asterisk character appears on the display to show that the option is unavailable.

20=Start Channel
Enables you to start an MQM channel.

21=End Channel
Enables you to end an MQM channel.

22=Work with Channel Status
Enables you to work with the status of a channel.

23=Reset Channel
Enables you to reset the message sequence number of a specified MQM channel.

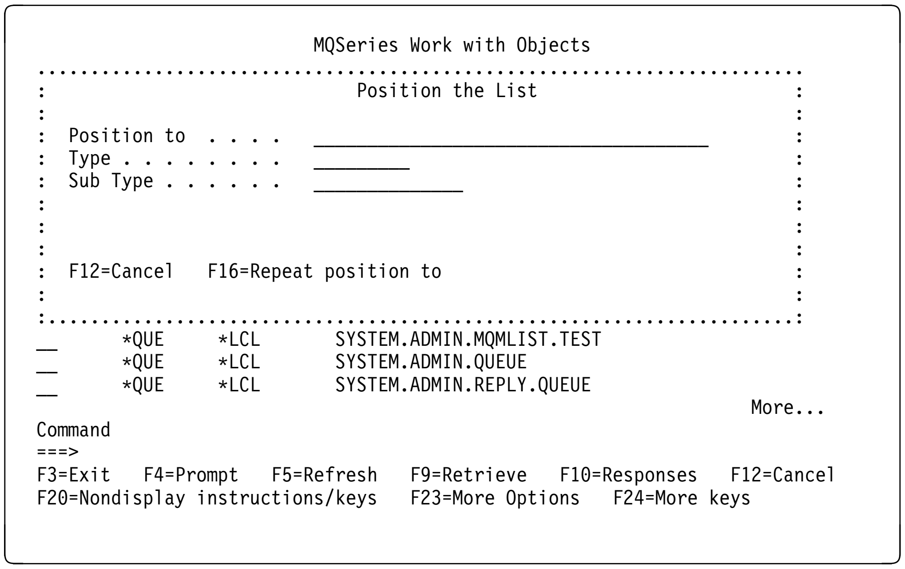
24=Start Channel Listener
Enables you to start the channel listener of a specified MQM channel.

25=Start Initiator
Enables you to start the channel initiator of a specified transmission queue.

26=Resolve Channel
Enables you to resolve a specified MQM channel.

Note: This option is only available if you are working with the local queue manager. If you are working with a remote queue manager, an asterisk character appears on the display to show that the option is unavailable.

30=Ping Channel
Enables you to test the operation of a specified MQM channel.



```
                         MQSeries Work with Objects
 ..................................................................
 :                           Position the List                     :
 :                                                                 :
 :  Position to  . . . .   _____________________________________   :
 :  Type . . . . . . . .   _________                               :
 :  Sub Type . . . . . .   ______________                          :
 :                                                                 :
 :                                                                 :
 :                                                                 :
 :  F12=Cancel   F16=Repeat position to                            :
 :                                                                 :
 :.................................................................:
 __      *QUE     *LCL       SYSTEM.ADMIN.MQMLIST.TEST
 __      *QUE     *LCL       SYSTEM.ADMIN.QUEUE
 __      *QUE     *LCL       SYSTEM.ADMIN.REPLY.QUEUE
                                                                  More...
 Command
 ===>
 F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Responses   F12=Cancel
 F20=Nondisplay instructions/keys   F23=More Options   F24=More keys
```

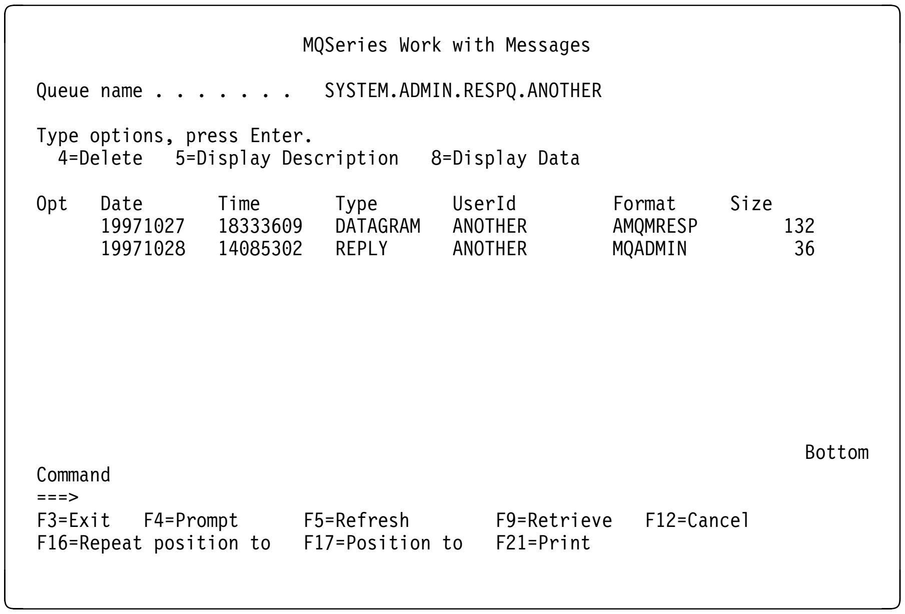
Figure 14. Position the List panel

Work with messages

This panel enables you to perform operations on messages in a specified local queue. Messages can be deleted and their contents or descriptions displayed.

The list of messages, shown on the selected queue, is valid only at the instant this panel is displayed. The contents of the queue may change while this list is being viewed.

Work with Messages is also available as the CL command, WRKMQMMSG. See "WRKMQMMSG (Work with MQM Messages) Command" on page 235 for a detailed explanation of the WRKMQMMSG command.



```
                          MQSeries Work with Messages

 Queue name . . . . . . .   SYSTEM.ADMIN.RESPQ.ANOTHER

 Type options, press Enter.
   4=Delete   5=Display Description   8=Display Data

 Opt   Date       Time       Type       UserId         Format     Size
       19971027   18333609   DATAGRAM   ANOTHER        AMQMRESP        132
       19971028   14085302   REPLY      ANOTHER        MQADMIN          36








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

Figure 15. MQSeries Work with Messages panel

Options for Work with Messages

4=Delete
This option allows you to delete the selected messages from the list.

5=Display Description
This option takes you to the Display MQSeries Message Description panel and enables you to display the message descriptor.

This function does not hold the message. Therefore, in the time between selecting the message and displaying its contents the message may disappear, that is, an application gets the message from the queue.

8=Display Data
This option takes you to the Display MQSeries Message Data panel and enables you to display the data in a message.

You should exercise care when displaying messages, if the messages concerned contain large amounts of data, due to the time taken to format the data.

Notes:

1. The queue name is the name of the selected queue.
2. `Opt` enables you to work with one of the available options.
3. `Date` is the date of the original MQM message, displayed as a string of eight digits in the format `YYYYMMDD`, where:
   - `YYYY` represents the year
   - `MM` represents the month
   - `DD` represents the date in the month

4. `Time` is the time of the original MQM message, displayed as a string of eight digits in the format `HHMMSSFF`, where:
   - `HH` represents the hour in 24 hour format
   - `MM` represents the minutes
   - `SS` represents the seconds
   - `FF` represents the hundredths of seconds
5. `Type` is the type of the message, which can be one of the following:
   - Datagram
   - Reply
   - Report
   - Request
6. `UserId` is the ID of the user sending the message.
7. `Format` is the format of the message, which can be one of the following:

`BLANK`	is displayed for	`NONE`
`MQADMIN`	is displayed for	`ADMIN`
`MQCHCOM`	is displayed for	`CHANNEL_COMPLETED`
`MQDEAD`	is displayed for	`DEAD_LETTER_HEADER`
`AMQMRESP`	is displayed for	`RESPONSE`
`MQSTR`	is displayed for	`STRING`
`MQTRIG`	is displayed for	`TRIGGER`
`MQXMIT`	is displayed for	`XMIT_Q_HEADER`

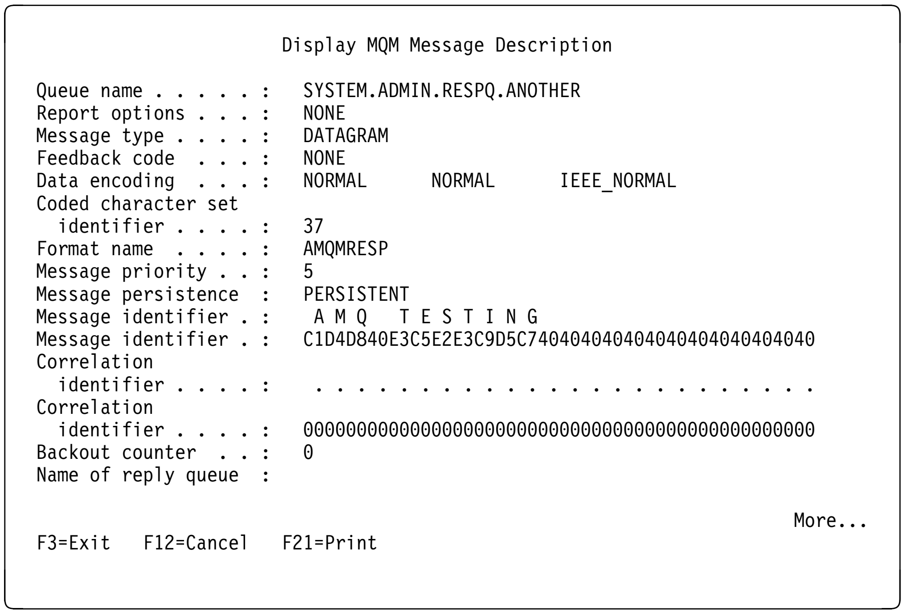
   Formats with the prefix `AMQM` are internal to the MQSeries for AS/400 Administration utility.
8. `Size` is the size of the message in bytes.

   A suffix of + indicates that all of the data has not been read into storage. See “WRKMQMMSG (Work with MQM Messages) Command” on page 235 for details of the WRKMQMMSG command.
9. Time, date, and User ID information only appears when the message contains contextual information.

Display Message Description

This panel displays, for a selected message, what each of the fields in the message descriptor header contains. For details of the message descriptor (MQMD), see the *MQSeries Application Programming Reference* manual or the *MQSeries for AS/400 Application Programming Reference (RPG)*, as appropriate.

Note: The dead-letter header of a message on the undelivered-message queue can be viewed only by looking at the user data, using Display MQM Message Data.



```
                        Display MQM Message Description

Queue name . . . . . :   SYSTEM.ADMIN.RESPQ.ANOTHER
Report options . . . :   NONE
Message type . . . . :   DATAGRAM
Feedback code  . . . :   NONE
Data encoding  . . . :   NORMAL      NORMAL      IEEE_NORMAL
Coded character set
  identifier . . . . :   37
Format name  . . . . :   AMQMRESP
Message priority . . :   5
Message persistence  :   PERSISTENT
Message identifier . :    A M Q   T E S T I N G
Message identifier . :   C1D4D840E3C5E2E3C9D5C740404040404040404040404040
Correlation
  identifier . . . . :    . . . . . . . . . . . . . . . . . . . . . . . .
Correlation
  identifier . . . . :   000000000000000000000000000000000000000000000000
Backout counter  . . :   0
Name of reply queue  :

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

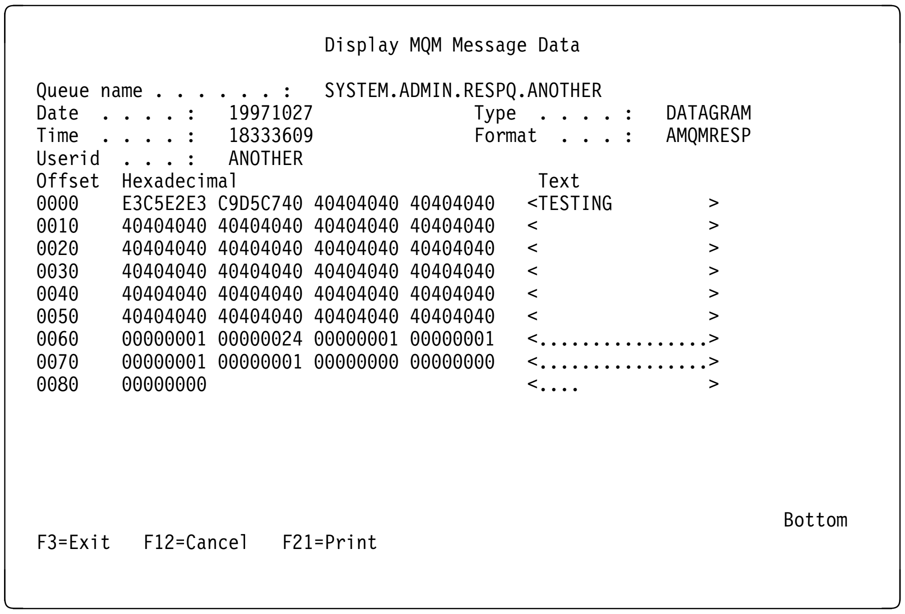
Figure 16. Display MQM Message Description panel

Fields containing data that is not character data are displayed as hexadecimal values, and character values where the character can be displayed.

Display Message Data

This panel displays the contents of a message. This operation does not hold the message. Therefore, in the time between selecting the function and displaying its contents, another application may remove the message from the queue.

The details on the display can be printed.



```
                            Display MQM Message Data

Queue name . . . . . . :   SYSTEM.ADMIN.RESPQ.ANOTHER
Date  . . . . :   19971027              Type  . . . . :   DATAGRAM
Time  . . . . :   18333609              Format  . . . :   AMQMRESP
Userid  . . . :   ANOTHER
Offset  Hexadecimal                           Text
0000    E3C5E2E3 C9D5C740 40404040 40404040   <TESTING         >
0010    40404040 40404040 40404040 40404040   <                >
0020    40404040 40404040 40404040 40404040   <                >
0030    40404040 40404040 40404040 40404040   <                >
0040    40404040 40404040 40404040 40404040   <                >
0050    40404040 40404040 40404040 40404040   <                >
0060    00000001 00000024 00000001 00000001   <................>
0070    00000001 00000001 00000000 00000000   <................>
0080    00000000                              <....            >




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

Figure 17. Display MQM Message Data panel

Notes:

1. The queue name is the name of the selected queue.
2. See "Options for Work with Messages" on page 55 for details on the following fields:
   - `Date`
   - `Time`
   - `UserId`
   - `Type`
   - `Format`
3. The following information is displayed about each message:

Offset	The offset from the start of the message data in hexadecimal notation of the first byte on the line.
Hexadecimal	The message data presented in hexadecimal notation.
Text	The printable characters corresponding to the previous hexadecimal notation.

Delete message confirmation

It is possible that a selected message may have been taken off the queue after the list has been displayed, and no longer exists. In this situation an error message is issued after the Delete Confirmation panel.

Note: You should exercise caution when deleting messages. If two messages displayed on the queue have the same message description and the displayed data is identical, the first message is deleted.

The meaning of "first," in this situation, is determined by the queue's message delivery attribute. To reduce the possibility of two identical messages appearing on the queue, you are recommended to put messages on the queue with the minimum context value being the default.

Work with responses

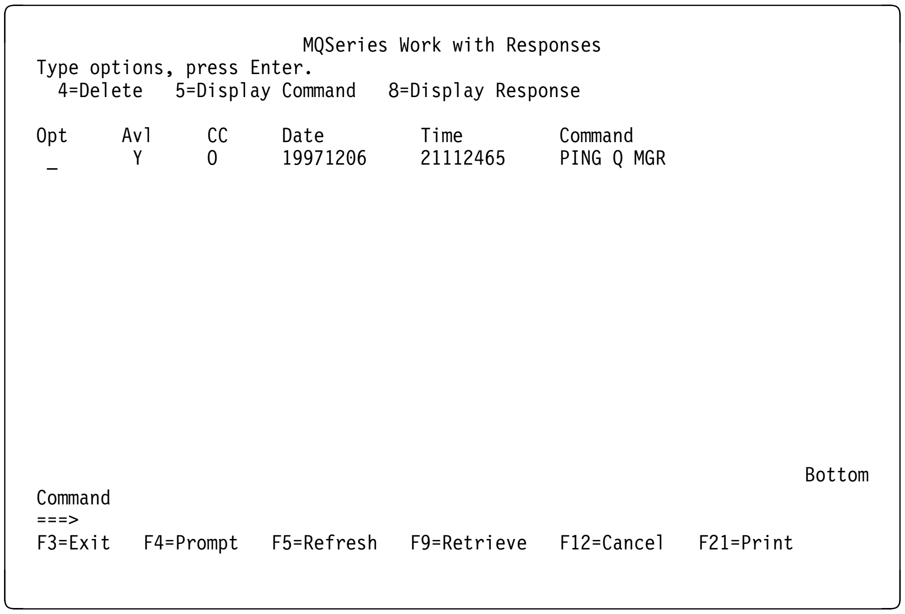
This panel shows you the status of operations for message queue managers. The responses to operations return in a time-independent manner to the invocation of the command.

You press Function key 10 (`PF10`) on the MQSeries Administration main panel, or the MQSeries Work with Objects panel, to display the MQSeries Work with Responses panel.

When you submit a command to a message queue manager using the Administration utility, the message queue manager sends a response back to the Administration utility to indicate whether or not the operation has been successful.

The responses return to the Administration utility in a time-independent manner, depending on the performance of the system and the network.

You use the MQSeries Work with Responses Panel to display commands that have been sent and their responses. When the availability of a response is `Yes`, and the completion code is `OK`, you know that the command has been successfully completed.



```
                          MQSeries Work with Responses
Type options, press Enter.
  4=Delete   5=Display Command   8=Display Response

Opt     Avl     CC     Date         Time         Command
 _       Y      O      19971206     21112465     PING Q MGR

                                                                            Bottom
Command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F12=Cancel   F21=Print
```

Figure 18. Response list

MQSeries Work with Responses panel

This panel allows you to select the following options:

4=Delete
Delete the response from the list.

If the response is incomplete, responses received after the selected response has been deleted, are discarded.

5=Display Command
Display the original command data. See Figure 19 on page 60.

8=Display Response
Display the response data (if available). See Figure 20 on page 61.

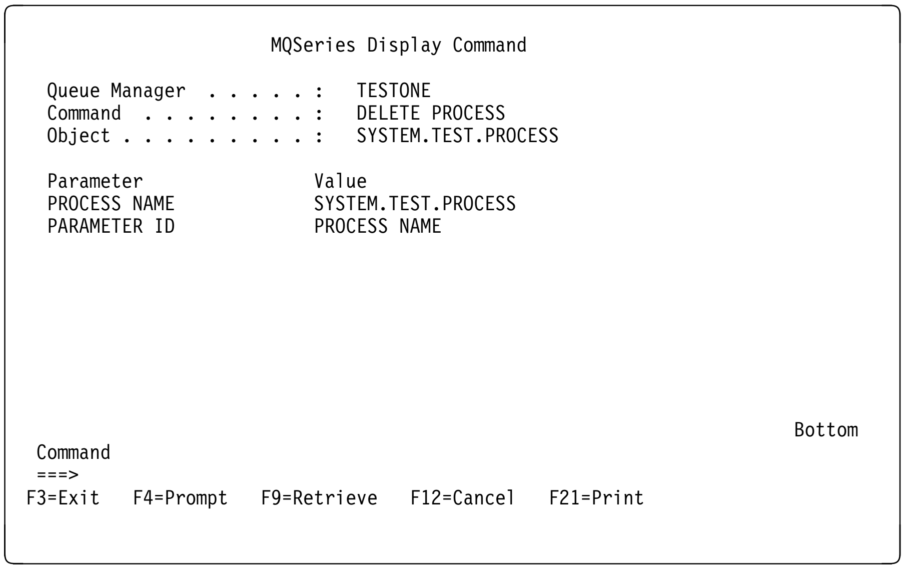
Notes:

1. `Avl`, the availability, can be:
   - Yes – available
   - No – not available
   - Partial – awaiting messages in the network
2. `CC`, the completion code, displays the maximum severity code received, where:

   `O` represents no problem
   `W` represents warning
   `F` represents failure
3. See “Options for Work with Messages” on page 55 for details on the following fields:
   - `Date`
   - `Time`
4. `Command` displays the name of the original command.

5. The date and time relate to the time the original MQM command message was sent.

Display Command



```
                     MQSeries Display Command

   Queue Manager  . . . . . :   TESTONE
   Command  . . . . . . . . :   DELETE PROCESS
   Object . . . . . . . . . :   SYSTEM.TEST.PROCESS

   Parameter               Value
   PROCESS NAME            SYSTEM.TEST.PROCESS
   PARAMETER ID            PROCESS NAME










                                                                 Bottom
  Command
  ===>
 F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F21=Print
```

Figure 19. MQSeries Display Command panel

This panel displays the selected command that you are working with, its associated parameter, or parameters, and their associated values.

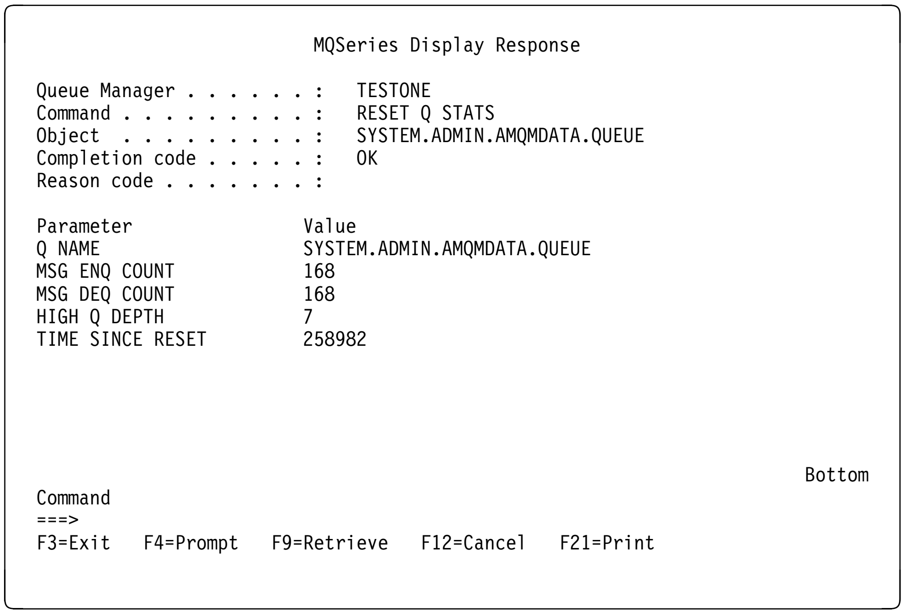
Notes:

1. The parameters are direct mappings of *values* shown in the MQSeries for AS/400 manual for the relevant programming language (the *MQSeries Application Programming Reference* manual or the *MQSeries for AS/400 Application Programming Reference (RPG)*). For example, `PROCESS NAME` is derived from `MQCA_PROCESS_NAME`.
2. If the parameter field displays `PARAMETER ID`, the value is taken from the appropriate include file. In the example shown, the value itself is `PROCESS NAME`.

Display Response data

This panel displays the response information, consisting of the completion code and associated reason code, for the selected remote command.

Additional data may be displayed as pairs of parameter names and associated values, if applicable.



```
                             MQSeries Display Response

Queue Manager . . . . . . :   TESTONE
Command . . . . . . . . . :   RESET Q STATS
Object  . . . . . . . . . :   SYSTEM.ADMIN.AMQMDATA.QUEUE
Completion code . . . . . :   OK
Reason code . . . . . . . :

Parameter                 Value
Q NAME                    SYSTEM.ADMIN.AMQMDATA.QUEUE
MSG ENQ COUNT             168
MSG DEQ COUNT             168
HIGH Q DEPTH              7
TIME SINCE RESET          258982





                                                                              Bottom
Command
===>
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F21=Print
```

Figure 20. MQSeries Display Response panel

Chapter 6. Recovery

MQSeries for AS/400 utilizes the OS/400 journaling support to aid in its backup and restore strategy. You should be familiar with standard AS/400 backup and recovery methods, and with the use of journals and their associated journal receivers on AS/400 before reading this section. For information on these topics, see the *AS/400 Backup and Recovery* book.

To understand the backup and recovery strategy, you should first understand how MQSeries for AS/400 organizes its data in the OS/400 file system.

MQSeries for AS/400 holds its data in two separate AS/400 libraries:

- `QMQMDATA`, which holds the majority of MQSeries for AS/400 data, including all queues and journal receivers.
- `QMQMPROC`, which holds the MQSeries process objects.

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.

Wherever possible, MQSeries objects are stored using the name by which the object is known to the user. This is not always possible, because not all MQSeries objects have valid AS/400 names. To allow for this situation, MQSeries for AS/400 reserves all names beginning with `Q`. If an MQSeries for AS/400 object does not have a valid OS/400 name, or if it has a name beginning with `Q`, it is assigned a unique name generated by MQSeries for AS/400 of the form `Qxxxxxxxxx`.

MQSeries for AS/400 journals

MQSeries for AS/400 uses two separate journals in its operation, one to control updates to local objects and one to control the flow of messages over communication links. These journals exist in the `QUSRSYS` library as:

QUSRSYS/AMQAJRN	MQSeries for AS/400 local journals detailing updates to local objects.
QUSRSYS/AMQRJRN	MQSeries for AS/400 remote journals detailing channel usage.

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.

They 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 becomes too large, it can be detached and replaced by a new journal receiver.

The journal receivers associated with the local MQSeries for AS/400 journal exist in `QMQMDATA`, and adopt a naming convention as follows:

```
AMQArnnnnn
```

where

nnnnn	is decimal 00000 to 99999
r	is decimal 0 to 9

From OS/400 V4R2 the MQSeries AMQRJRN journal continues to use *USER management of journal receivers but the AMQAJRN journal uses *SYSTEM. Therefore, for AMQAJRN, the operating system manages the changing of the journal receivers.

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 AMQA099999 goes to AMQA100000.
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 will need to do both the following:
   a. Delete journals no longer needed (see "Journal management" on page 67).
   b. Record the journal changes into the latest journal receiver using (RCDMQMIMG) and then repeat the previous step. This will allow the old journal receiver names to be reused.

By default, MQSeries specifies a threshold of 16384 KB. If this value is insufficient, you can increase the threshold value by creating and attaching a new journal receiver with the value you require.

The journal receivers associated with the remote MQSeries for AS/400 journal also exist in QMQMDATA, and adopt a similar naming convention as follows.

```
AMQRrnnnnn
```

where

nnnnn	is decimal 00000 to 99999
r	is decimal 0 to 9

The remote journals are used to control updates to the channel synchronization file `QMQMDATA/AMQRSYNA`. All updates to this file are journaled to the remote journal.

There is no automatic change-journal management for remote receivers. It is your responsibility to monitor the size of the AMQR* journal receivers in the QMQMDATA library and attach a new journal receiver as required.

See “Journal management” on page 67 for details on how to manage these journal receivers.

MQSeries for AS/400 local journal usage

To understand how MQSeries for AS/400 uses the local journal, consider the case of a local queue called `TESTQ`, which is an MQSeries for AS/400 queue that has a valid OS/400 name not beginning with “Q”. This is represented by the `*USRSPC` object `QMQMDATA/TESTQ`.

If a specified message is put on this queue, and then retrieved from the queue, the actions that take place are shown in Figure 21.

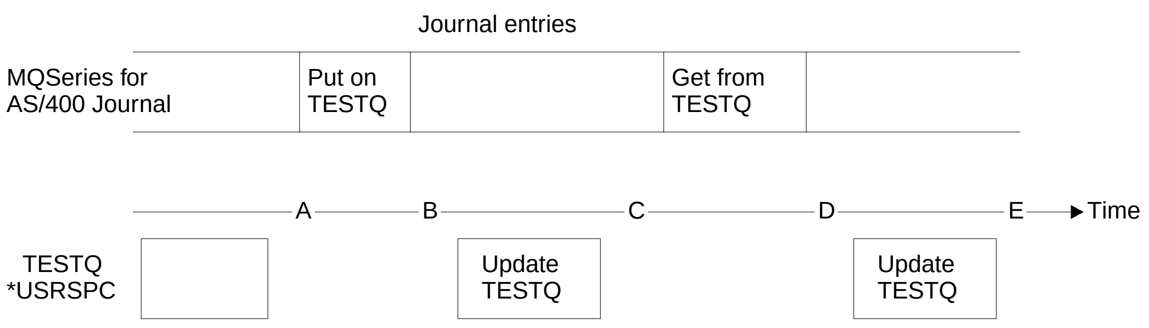


Figure 21. Sequence of events when updating MQM objects

The five points, “A” through “E,” shown in the diagram represent points in time that define the following states:

A The `*USRSPC` 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 MQSeries for AS/400 is that the user can save the USRSPC representation of TESTQ as at time `A`, and subsequently recover the USRSPC 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 MQSeries for AS/400 to provide recovery of persistent messages after system failure. MQSeries for AS/400 remembers a particular entry in the journal receivers, and ensures that on startup it will replay the entries in the journals from this point onwards. This startup entry is periodically recalculated so that MQSeries for AS/400 only has to perform the minimum necessary replay on the next startup.

MQSeries for AS/400 provides individual recovery of objects. All persistent information relating to an object is recorded in the local MQSeries for AS/400 journals. Any MQSeries for AS/400 object that becomes damaged or corrupt can be completely rebuilt from the information held in the journal.The local journals use system change-journal management (MNGRCV(*SYSTEM)), which means that the system controls the switching of journal receivers. If you need to alter the journal receiver settings, for example, to increase the journal receiver threshold size, create a journal receiver with your preferred settings and manually attach it. All subsequent receivers will be created using your preferred settings.

For more information on how the system manages receivers, see the *AS/400 Backup and Recovery* book.

Media images

For an MQSeries for AS/400 object of long duration, this can represent a large number of journal entries, going back to the point at which it was created. To avoid this overhead, MQSeries for AS/400 has the concept of a *media image* of an object.

This media image is a complete copy of the MQSeries for AS/400 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 MQSeries for AS/400 object is referred to as its *media recovery entry*.

Images of the three important MQM objects, that is, the *CTLG object, the *ADM object, and the *MQM object, are regularly taken because these objects are required for MQSeries for AS/400 to run at all.

Images of other objects are taken when convenient, particularly when the MQSeries for AS/400 queue manager is ended. MQSeries for AS/400 keeps track of the:

- Media recovery entry for each MQM object
- Oldest entry from within this set

MQSeries for AS/400 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 may never happen for some objects, for example, queues which consistently contain large numbers of messages.

Rather than allow the date of the oldest media recovery entry to continue for an unnecessarily long period, you should use the MQSeries for AS/400 command RCDMQMIMG This command enables you to take an image of selected objects manually; see “RCDMQMIMG (Record MQM Object Image) Command” on page 207 for further details of this command.

Recovery from media images

MQSeries for AS/400 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 *CTLG, *ADM and *MQM objects, if they are damaged 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 MQSeries for AS/400 command `RCRMQMOBJ`; see “RCRMQMOBJ (Recreate MQM Object) Command” on page 209 for details of this command.

This command replays the entries in the journal to recreate the MQSeries object. Should an MQSeries object become damaged, the only valid actions that may be performed are to delete it or to re-create it by this method. Note, however, that nonpersistent messages cannot be recovered in this fashion.

Note: The authorities of the recreated object are *not* reapplied by this method. The appropriate authorities *must* be manually set after the object has been recreated. When an object is recreated, message AMQ7461 is sent to the system operator as a reminder to recreate the object authorities.

Recovery of the channel synchronization file must be done manually, using the OS/400 APYJRNCHG command, and applying the changes recorded in the remote MQSeries for AS/400 journals to the synchronization file.

Backups of MQSeries for AS/400 data

There are two general types of MQSeries backup that should be considered:

Full backup

This involves a full backup of the two MQSeries for AS/400 libraries and should be performed by issuing the following AS/400 SAVLIB command:

```
SAVLIB LIB(QMQMDATA QMQMPROC) .........
```

A save-while-active request cannot complete unless all commitment definitions with pending changes are committed or rolled back. Therefore, if this command is used when there are active MQSeries channels, the channel connections may end abnormally.

Before using the SAVLIB command, you are recommended to carry out the following procedure:

1. End all MQSeries channels
2. Use the RCDMQMIMG command to record an MQM image for all MQSeries objects
3. Quiesce MQSeries, including running the MQSeries program AMQIQEM4. This significantly reduces the amount of data to be saved in library QMQMDATA.

Journal backup

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 should be performed by issuing the following command:

```
SAVOBJ OBJ(AMQ*) LIB(QMQMDATA) OBJTYPE(*JRNRCV) .........
```

A simple backup strategy is to perform a full backup of the MQSeries for AS/400 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, you should take care of your journal receivers. It is useful to remove journal receivers from the MQSeries for AS/400 libraries, in order to:

| 1. Release space – applies to all journal receivers

2. Improve the performance when starting (STRMQM) – applies to local journal receivers only
3. Improve the performance of recreating objects (RCRMQMOBJ) – applies to local journal receivers only

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 library QMQMDATA *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 22.

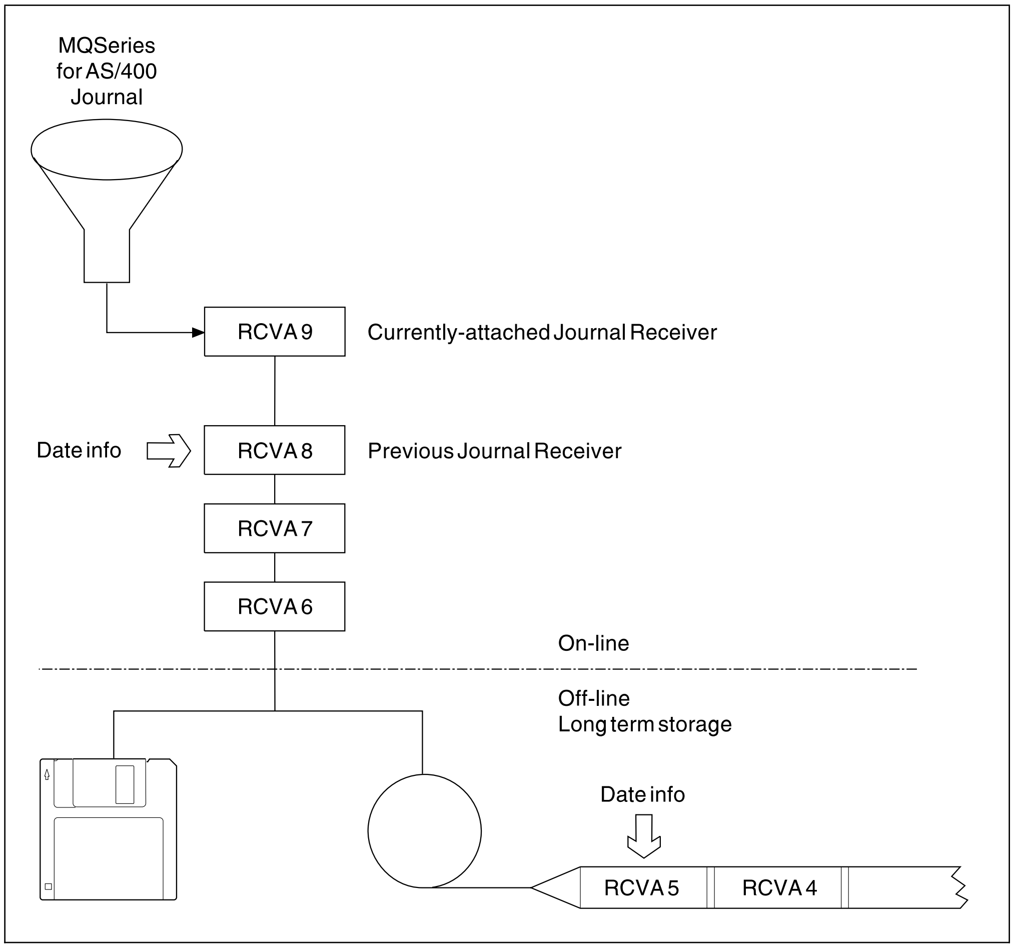

MQSeries for AS/400 Journal
RCVA 9
Currently-attached Journal Receiver
Date info
RCVA 8
Previous Journal Receiver
RCVA 7
RCVA 6
On-line
Off-line
Long term storage
Date info
RCVA 5
RCVA 4

Figure 22. MQSeries for AS/400 journaling

It is important to know how far back in the journals MQSeries for AS/400 is likely to need to go, in order to determine when a journal receiver that has been backed up may be removed from the library QMQMDATA, and when the backup itself may be discarded.

To help determine this time, MQSeries for AS/400 issues two messages to the system operator queue whenever it starts up, and whenever it changes a local journal receiver. These messages are:

AMQ7460 Startup recovery point. This message defines the date and time of the startup entry from which MQSeries for AS/400 replays the journal in the event of a startup recovery pass. If the journal receiver that contains this record is available in the MQSeries for AS/400 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 that may be used for recreating an object from its media image.

| **Note:** Periodically performing `RCDMQMIMG(*ALL)` can save startup time for MQSeries and reduce the number of local journal receivers you need to save and restore for recovery.

| MQSeries for AS/400 does not refer to the local 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` reporting the time and date of the journal entry it requires to complete the recovery pass.

If this happens, all local journal receivers that were detached after this date should be restored from the backup, in order to allow the recovery pass to succeed.

| The journal receiver that contains the startup entry, and any subsequent journal receivers should be kept available in the library QMQMDATA.

| The journal receiver containing the oldest `Media Recovery Entry`, and any subsequent journal receivers, should be available at all times, and either present in library QMQMDATA or backed-up.

| MQSeries for AS/400 does not carry out any automatic management of the remote journal receiver, AMQRJRN. Either you perform the management, or OS/400 performs the management using the `MNGRCV *SYSTEM` option. A message queue, `QMQMDATA/AMQRJRNMSG`, is associated with the journal receiver and this is initially attached for the remote journal. However, it is your responsibility to monitor journal messages on this queue.

| For the remote journal there is only one file journaled, and this is the remote synchronization file `QMQMDATA/AMQRSYNA`. This means that the only recovery point is when the object was last saved.

| Provided the AMQRSYNA file has been saved since the current remote journal receiver was generated, you can safely delete all the earlier remote receivers. You can perform this operation at any time, irrespective of whether MQSeries channels are running.

Restoring journal receivers

| The most common action is to restore a backed-up journal receiver to library QMQMDATA, 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 AS/400 RSTOBJ command:

```
RSTOBJ OBJ(QMQMDATA/AMQA000005) OBJTYPE(*JRNRCV) .........
```

It may be that a series of journal receivers needs to be restored, rather than simply a single receiver. For example, `AMQA000007` is the oldest receiver in the MQSeries for AS/400 libraries, and both `AMQA000005` and `AMQA000006` need to be restored.

In this case the receivers should be restored individually in reverse chronological order. This is not always necessary, but is good practice. In severe situations, the OS/400 command `WRKJRNA` may be required, in order 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 may 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 QUSRSYS, the system automatically creates journal receiver AMQA000000. This automatically generated receiver conflicts with one of the existing journal receivers (AMQA000000) you wish to restore, which you will be unable to restore.

The solution is:

1. Manually create the next journal receiver (see “MQSeries for AS/400 journals” on page 63).

   ```
   CRTJRNRCV JRNRCV(QMQMDATA/AMQA900001) MNGRCV(*SYSTEM)
   THRESHOLD(16384)
   TEXT('MQM LOCAL JOURNAL RECEIVER')
   ```

2. Manually create the journal with the above journal receiver:

   ```
   CRTJRN JRN(QUSRSYS/AMQAJRN)
   JRNRCV(QMQMDATA/AMQA9000001)
   MSGQ(QMQMDATA/AMQAJRNMSG)
   TEXT('MQM local journal')
   ```

3. Restore the local journal receivers AMQA000000 - AMQA900000.

Restoring a complete system

If you need to recover a complete MQSeries for AS/400 system from a backup you should perform the following steps.

1. Quiesce the MQSeries for AS/400 system; see “Quiescing MQSeries for AS/400” on page 16.
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 MQSeries for AS/400 data libraries to their state at the time of the full backup, by issuing the following commands:

   ```
   RSTLIB LIB(QMQMDATA) .........
   RSTLIB LIB(QMQMPROC) .........
   ```

4. Perform a RSTOBJ operation, from the journal backups, to restore the MQSeries journals AMQAJRN, and AMQRJRN to QUSRYS.

If a journal receiver was partially saved in one journal backup, and fully saved in a subsequent backup, the fully saved one only should be restored. You are recommended to restore journals individually, in chronological order.

5. Recover the remote queue manager synchronization file by working with the OS/400 APYJRNCHG command.
6. Start the message queue manager. This will replay all journal records written since the full backup and restores all the MQSeries for AS/400 objects to the consistent state at the time of the journal backup.

Chapter 7. Analyzing problems

This chapter suggests reasons for problems you may have with MQSeries for AS/400. This process is called problem determination. You usually start with a symptom, or set of symptoms, and trace them back to their cause.

You should not confuse problem determination with problem solving; however, 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 may not always be able to solve a problem after determining its cause. For example:

- A performance problem may be caused by a limitation of your hardware.
- You may find that the cause of the problem is in the MQSeries for AS/400 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

Before you start problem determination in detail, it is worth considering the facts to see if there is an obvious cause of the problem, or an area likely, in 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
- MQSeries product
- Your MQSeries application
- Other applications
- Site operating procedures

The sections that follow raise some fundamental questions that you will 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.

© Copyright IBM Corp. 1994, 1998

The following steps are intended to help you isolate the problem and are taken from the viewpoint of an MQSeries application. You are recommended to check all the suggestions at each stage.

1. Has MQSeries for AS/400 run successfully before?

   Yes
   Proceed to Step 2.

   No
   It is likely you have not installed or setup MQSeries correctly, see "Standard AS/400 installation procedure" on page 15.

2. Has the MQSeries application run successfully before?

   Yes
   Proceed to Step 3.

   No
   Consider the following:

   a. The application may 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 *MQSeries 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 MQSeries 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 been 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 appropriate *MQSeries Application Programming Reference* for your programming language for a description of each return code.

d. If you have applied any PTF to MQSeries for AS/400, 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 AS/400 system remain 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 may have impacted the MQSeries 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 should now be able to resolve it, possibly with the help of other books in the MQSeries library (see “MQSeries publications” on page x) and in the libraries of other licensed programs.

If you have not yet found the cause, you must start to look at the problem in greater detail. The following questions should be used as pointers to the problem. Answering the appropriate question, or questions, should lead you to the cause of the problem.

Can the problem be reproduced?

If the problem is reproducible, consider the conditions under which it can be reproduced:

- Is it caused by a command or an equivalent Administration utility request?

  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 MQSeries for AS/400 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.

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 may 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).

Does the problem affect all users of the MQSeries for AS/400 application?

If the problem only affects some users, is this because some users do not have the correct security authorization? See “MQSeries for AS/400 security considerations” on page 23 for further information about MQSeries for AS/400 security.

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 that the connection between the two systems is available, and that the intercommunication component of MQSeries for AS/400 has been started.)

Check that messages are reaching the transmission queue, and check 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 MQSeries for AS/400 definitions?

Does the problem occur on a specific version of the operating system?

If the problem occurs on a specific version of MQSeries, for example, MQSeries for OS/400 V3R7 or MQSeries for AS/400 V4R2M1, check the appropriate database on RETAIN 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 midmorning and midafternoon, and so these are the times when load-dependent problems are most likely to occur. (If your MQSeries for AS/400 network extends across more than one time zone, peak system loading might seem to occur at some other time of day.)

If you have still not identified the cause of the problem, see “Determining problems with MQSeries applications.”

Determining problems with MQSeries applications

This section discusses problems you may encounter with MQSeries applications, commands, and messages.

Have you failed to receive a response from a command?

If you have issued a command, or Administration utility request, 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 undelivered- message (dead-letter) queue?

  Work with the DSPMQMQ command, for the undelivered-message queue, to see if there are any messages on the queue.

  The dead-letter queue header structure, which is added in front of the user-data area contains a reason or feedback code describing the problem. The structure is found by looking at the message contents. See the *MQSeries Application Programming Reference* manual or the *MQSeries for AS/400 Application Programming Reference (RPG)*, as appropriate, for information about the undelivered-message (dead-letter) header structure (MQDLH).

- Has a message been sent to `QSYSOPR`? Use the DSPMSG `QSYSOPR` command to display these messages.

- Are the correct queues defined, that is, the `SYSTEM.ADMIN.COMMAND.QUEUE` and those required by the Administration utility (see “Setting up the Administration utility” on page 43).

- Are the queues enabled for put and get operations?

- Is the *`WaitInterval`* set to a sufficient length?

  If your MQGET call has timed out, you will see a completion code of 2 and a reason code of 2033 (MQRC_NO_MSG_AVAILABLE). (See the *MQSeries Application Programming Reference* manual or the *MQSeries for AS/400 Application Programming Reference (RPG)*, as appropriate, for information about the *WaitInterval* field, and completion and reason codes from the MQGET call.)

- 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 will need to take a syncpoint before attempting to receive reply messages.
- Are the MAXDEPTH and MAXMSGLEN 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 the whole of the MQSeries for AS/400 system. Try stopping and restarting MQSeries for AS/400, responding to any messages that are produced in the joblog.

If the problem still occurs after restart, contact your IBM support center for help.

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.
2. Use the data displayed to do the following checks:
   - If CURDEPTH is at MAXDEPTH this indicates that 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 are unable to 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 the *MQSeries Intercommunication* book for information about how to do this.
4. Check the channel with a PING command.

See the *MQSeries Intercommunication* book for information about how to define channels.

Does the problem affect messages?

This section deals with:

- "Messages do not appear on the queue"
- "Messages contain unexpected or corrupted information" on page 80
- "Receiving unexpected messages when using distributed queues" on page 81

Messages do not appear on the queue

If messages do not appear when you are expecting them, check for the following:

- Has the message been put on the queue successfully?
  - Has the queue been defined correctly, for example is MAXMSGLEN sufficiently large?
  - Are applications able to 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.
- Are you able to 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 or correlation identifier (*MsgId* or *CorrelId*)?

    Check that you are waiting for a message with the correct *MsgId* or *Correlid*. A successful MQGET call will set both these values to that of the message retrieved, so you may 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 MQSeries for AS/400 has been restarted, the message will have been lost.

If you are unable to find anything wrong with the queue, and MQSeries for AS/400 itself is running, make the following checks on the process that you expected to put the message on to the queue:

- Did the application get started?

  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

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 may have been changed, in which case, both applications will have to be recompiled to pick up the changes. If one application has not been recompiled, the data will appear 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?

If these checks do not enable you to solve the problem, you should check your application logic, both for the program sending the message, and for the program receiving it.

Receiving unexpected messages when using distributed queues

If your application uses distributed queues, you should also 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 MQSeries for AS/400. 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) will contain a reason or feedback code explaining why the message could not be put on to the target queue. See the *MQSeries Application Programming Reference* manual or the *MQSeries for AS/400 Application Programming Reference (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 the *MQSeries Intercommunication* book for more information about distributed queuing.

Obtaining diagnostic information

This sections tells you where to find diagnostic information about MQSeries.

1. **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 MQSeries commands issued and the sequence of those commands.
2. **MQSeries Job Log:** MQSeries specific jobs, for example, the command server and channel programs, run under the MQSeries profile QMQM. If you have a problem in these areas, review these joblogs by issuing the command `WRKSPLF QMQM` to display them.

See Appendix C, “Specific MQSeries for AS/400 considerations” on page 273 for the names of specific MQSeries jobs.

3. **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.
4. **AS/400 Message Queue:** It is useful to view messages sent to various AS/400 message queues using the DSPMSG command. Use the command `DSPMSG QSYSOPR` to check the system operator message queue, used for MQSeries journalling messages, and job completion messages in particular.
5. **Work with Problems:** Use the WRKPRB command to display descriptions of system problems. MQSeries reports problems related to unusual usage and internal code by using this command.

Using MQSeries for AS/400 trace

Although it will be necessary to use certain traces on occasion, running the trace facility will slow your systems.

You should also consider to what destination you want your trace information sent.

Notes:

1. To run the MQSeries for AS/400 trace commands, you must have the appropriate authority, or have one of the user IDs specified in “Security considerations for tracing MQSeries for AS/400” on page 25.
2. Trace data is only written when trace is ended, with option *OFF

Lifetime of trace data

Trace data remains in the system until you delete it by issuing the command:

- TRCMQM **END* or,
- STRMQMSRV, with the relevant job name, user, and job number, to connect to the historical trace data, followed by TRCMQM **END*.

Unless you delete it, the trace data remains until the storage monitor is ended, at which point the trace files are written out to the `QMQM` spool.

Trace usage

A trace can be obtained using TRCMQM **ON*, doing some MQ work, and TRCMQM **OFF* at the end of the MQ work being traced.

Batch jobs inherit the trace attributes of the calling program, but if you have not started trace in the calling program, you can use STRMQMSRV followed by the TRCMQM *ON* and TRCMQM **OFF* calls. If the job ends before you issue the TRCMQM **OFF* calls, you can connect to it from another job using STRMQMSRV, and then issuing TRCMQM **OFF*.

Setting trace before you issue the command that starts a batch job, has the advantage that the batch job will be traced from its start.

Selective trace

By default, TRCMQM traces all MQSeries product components saving the maximum amount of trace data and using the *WRAP option in cases where the trace file becomes full.

You can reduce the amount of trace data being saved, thereby improving run-time performance, using the command `TRCMQM *ON` with `F4=prompt` to customize the TRCTYPE and TRCCOMP parameters.

TRCTYPE specifies the type of data to store in the trace file:

***API**	Trace entry and exit at the API level
***COMP**	Trace entry and exit at the component level
***FUNC**	Trace entry and exit at the function level
***MSG**	Messages issued by MQSeries are stored, trace data is message identifier and CCSID only
***COMM**	Communications data buffers
***ALL**	All of the above

TRCCOMP specifies the scope of tracing by component:

***ADM**	The Administrator utility component is traced.
***AIC**	The Application Interface component is traced.
***CMD**	The MQSeries for AS/400 commands are traced.
***COM**	The Communications component is traced.
***CSR**	The Command Server component is traced.
***CSV**	The Common Services component is traced.
***DCV**	The Data Conversion component is traced.
***DHD**	The Data Hardening component is traced.
***INS**	The Installation component is traced.
***LMT**	The Log Management component is traced.
***LQK**	The Local Queue Manager kernel component is traced.
***LQM**	The Local Queue Manager component is traced.
***OAM**	The Object Authority component is traced.
***OCT**	The Object Catalog component is traced.
***QMT**	The Queue Management component is traced.
***RAS**	The Remote API Server component is traced.
***RQP**	The Remote Queue Processor component is traced.
***TMT**	The Transaction Management component is traced.
***ALL**	All of the above.

Some suggested combinations of components are :

Administration utility
(includes command server and data conversion) *ADM *CSR *DCV

Installation
*INS

Local Queue Manager
(applications and commands interface) *AIC *CMD *CSR *LQK *OAM

Local Queue Manager
(full, including object handling and log management) *AIC *CMD *CSR *DCV *DHD *LMT *LQK *LQM *OAM *OCT *QMT *TMT

Remote operations
(includes Message Channel Agent) *AIC *COM *CSR *DCV *RAS *RQP

Trace using MQSeries for AS/400 commands

Tracing batch jobs submitted by certain MQM commands:

- Set trace in the current job by issuing the command TRCMQM **ON*.
- Issue the command that submits the job.

  Trace is now active for both the current and submitted job.
- Issue the command TRCMQM **OFF* to extract trace for the current job. This ends trace for the current job, but trace is still active for the submitted job.
- To extract the trace for the submitted job:
  - Issue STRMQMSRV for the submitted job. You can find out the submitted job name by one of the following methods:
    1. If the job is still running, by using WRKACTJOB.
    2. If the job has finished, by looking at the trace for the current job. The submitted job identifier is displayed immediately after the entry for `xcsExecProgram`.
  - Issue the command TRCMQM **OFF* to act on the submitted job. If the trace output has been directed to **PRINT*, it is written to the current job's spool.
  - Issue the command ENDMQMSRV, which returns subsequent trace commands to act on the current job.

Note: If tracing both batch and interactive jobs, ensure that both traces have been ended with TRCMQM *OFF.

In addition, look in the spooled files, for user `QMQM`, for anything the storage monitor or others may have written.

Performance considerations

This section discusses:

- General design considerations - see “Application design considerations”
- Specific performance problems - see “Specific performance problems” on page 86

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 MQSeries for AS/400 calls are discussed in the following sections.

For more information about application design, see the *MQSeries Application Programming Guide*.

Effect of message length

Although MQSeries for AS/400 allows messages to hold up to 4 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, you should send only the essential data in a message; for example, in a request to debit a bank account, the only information that may 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 you should 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 as high as 4 MB, the maximum allowed by MQSeries for AS/400.

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.

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 user space associated with MQSeries for AS/400. Further details on how to clear some user space are given in "Quiescing MQSeries for AS/400" on page 16.

Is your application or MQSeries for AS/400 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 midmorning and midafternoon. (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, then 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 MQSeries for AS/400 is running slowly:

- If your system is slow to respond to MQSeries for AS/400 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 being run?

Chapter 8. The MQSeries 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. [1]

Queue managers, message channel agents, and applications can put messages on the DLQ. All messages on the DLQ should 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. You are strongly recommended to 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 MQSeries environments, there should be a routine that runs regularly to process messages on the DLQ. MQSeries 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/QTXTCSRC)
```

- In the rules table. For example:

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

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. Furthermore, you must be authorized to assume the identity of other users, if the

1 It is often preferable to avoid placing messages on a DLQ. For information about the use and avoidance of DLQs, see the *MQSeries Application Programming Guide*.

© Copyright IBM Corp. 1994, 1998

DLQ handler is to be able to put messages on queues with the authority of the user ID in the message context.

For more information about the STRMQMDLQ command, see "STRMQMDLQ (Start MQSeries Dead-Letter Queue Handler)" on page 223.

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. Please 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***|`'_'`)**
This parameter allows you to name the DLQ you want to process:

1. If you specify an UDLMSGQ value (or *DFT) as a parameter to the STRMQMDLQ command, this 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***|`'_'`)**
This parameter allows you to name 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***|<u>60</u>)**
This parameter is 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*)

This parameter indicates 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	This parameter 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.

You are recommended to specify WAIT (YES) for busy DLQs, and WAIT (NO) or WAIT (*nnn*) for DLQs that have a low level of activity. If the DLQ handler is allowed to terminate, you are recommended to reinvoke it by means of triggering.

You can supply the name of the DLQ as an input parameter of 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)

Figure 23 shows an example rule from a DLQ handler rules table.

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
 ACTION (RETRY) RETRY (3)
```

Figure 23. An example rule from a DLQ handler rules table. 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. Please 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. You use these to specify values against which messages on the DLQ are matched. All pattern-matching keywords are optional.

APPLIDAT (*ApplIdentityData***|*)**
This parameter is the *ApplIdentityData* value of the message on the DLQ, specified in the message descriptor, MQMD.

APPLNAME (*PutApplName***|*)**
This parameter is 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***|*)**
This parameter is the *PutApplType* value specified in the message descriptor, MQMD, of the message on the DLQ.

DESTQ (*QueueName***|*)**
This parameter is the name of the message queue for which the message is destined.

DESTQM (*QueueManagerName***|*)**
This parameter is 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***|*)**
This parameter is the name that the sender of the message uses to describe the format of the message data.

MSGTYPE (*MsgType***|*)**
This parameter is 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***|*)**
This parameter is 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***|*)**
This parameter is 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_**|*)**
This parameter is the reply-to queue name specified in the message descriptor, MQMD, of the message on the DLQ.

REPLYQM (_QueueManagerName_**|*)**
This parameter is the queue manager name, of the reply-to queue, specified in the REPLYQ keyword.

USERID (_UserIdentifier_**|*)**
This parameter is 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. You use these to describe how a matching message is processed.

ACTION (DISCARD|IGNORE|RETRY|FWD)
This describes 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)**
This parameter defines the name of the message queue to which the message is forwarded when you select the ACTION keyword.

QueueName	This parameter is the name of a message queue. FWDQ(' ') is not valid.
&DESTQ	Takes the queue name from the _DestQName_ field in the MQDLH structure.
&REPLYQ	Takes the name from the _ReplyToQ_ field in the message descriptor, MQMD. You can specify REPLYQ (?*) in the message pattern to avoid error messages, when a rule specifying FWDQ (&REPLYQ), matches a message with a blank _ReplyToQ_ field.

FWDQM (_QueueManagerName_**|&DESTQM|&REPLYQM|' _')**
Identifies the queue manager of the queue to which a message is forwarded.

QueueManagerName
This parameter defines the queue manager name, for the queue, to which the message is forwarded when you select the ACTION (FWD) keyword.

&DESTQM
Takes the queue manager name from the *DestQMgrName* field in the MQDLH structure.

&REPLYQM
Takes the name from the *ReplyToQMgr* field in the message descriptor, MQMD.

`' '`
FWDQM(`' '`), which is the default value, identifies the local queue manager.

HEADER (<u>YES</u>|NO)
Specifies 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 (<u>DEF</u>|CTX)
Defines 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*|<u>1</u>)
This parameter is the number of times, in the range 1–999 999 999, that an action should be attempted (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 reasons of portability, the significant length of a line should not be greater than 72 characters.
- Use the plus sign (+) as the last nonblank character on a line to indicate that the rule continues from the first nonblank character in the next line. Use the minus sign (–) as the last nonblank character on a line to indicate that the rule continues from the start of the next line. Continuation characters can occur within keywords and parameters.

  For example:

  ```
  APPLNAME('ABC+
    D')
  ```

  results in 'ABCD'.

  ```
  APPLNAME('ABC-
    D')
  ```

  results in 'ABC D'.
- Comment lines, which begin with an asterisk (*), can occur anywhere in the rules table.
- Blank lines are ignored.
- Each entry in the DLQ handler rules table comprises one or more keywords and their associated parameters. The parameters must follow these syntax rules:
  - Each parameter value must include at least one significant character. The delimiting quotation marks in 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.

Notes:

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 may 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 the *MQSeries Messages* book.) 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 may 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 could 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). Therefore, if the queue is not empty, the DLQ is periodically rescanned to check all messages. For these reasons, you should 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, the final rule in the table should be a catchall to process messages that have not been addressed by earlier rules in the table. For example, the final rule in the table could be something like this:

```
 ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

This 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 are 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 do 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 should 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)
```

Part 2. Reference

Chapter 9. Commands and utilities

This chapter contains reference material for the CL commands used with MQSeries for AS/400. All commands in this chapter :

- Are subjected to authorization checking
- Can be issued from a terminal

Additionally, for most of the commands, requests equivalent to the commands can be issued by a suitably-authorized local or remote application sending a message to a SYSTEM.ADMIN.COMMAND.QUEUE. The formats of these command messages and responses are documented in the *MQSeries Programmable System Management* manual.

Note: For command requests issued by an application, the responses are returned to the specified reply-to queue.

The commands, which are in alphabetic order in this chapter , can be grouped as follows:

- Administrator Command

  STRMQMADM, Start MQM Administrator (page 219)

- Channel Commands

  CHGMQMCHL, Change MQM Channel (page 112)
  CPYMQMCHL, Copy MQM Channel (page 134)
  CRTMQMCHL, Create MQM Channel (page 157)
  DLTMQMCHL, Delete MQM Channel (page 183)
  DSPMQMCHL, Display MQM Channel (page 190)
  ENDMQMCHL, End MQM Channel (page 197)
  PNGMQMCHL, Ping MQM Channel (page 206)
  RSTMQMCHL, Reset MQM Channel (page 211)
  RSVMQMCHL, Resolve MQM Channel (page 212)
  STRMQMCHL, Start MQM Channel (page 220)
  STRMQMCHLI, Start MQM Channel Initiator (page 221)
  STRMQMLSR, Start MQM Listener (page 225)
  WRKMQMCHL, Work with MQM Channel (page 232)
  WRKMQMCHST, Work with MQM Channel Status (page 233)

- Command Server Commands

  DSPMQMCSVR, Display MQM Command Server (page 191)
  ENDMQMCSVR, End MQM Command Server (page 198)
  STRMQMCSVR, Start MQM Command Server (page 222)

- Data Type Conversion Command

  CVTMQMDTA, Convert MQM Data Type Command (page 180)

| - Dead-Letter Queue Handler Command

| STRMQMDLQ, Start MQSeries Dead-Letter Queue Handler (page 223)

- Media Recovery Commands

  RCDMQMIMG, Record MQM Object Image (page 207)
  RCRMQMOBJ, Recreate MQM Object (page 209)

- MQSeries Command

STRMQMMQSC, Start MQSeries Commands (page 227)

- Name Command

  DSPMQMOBJN, Display MQM Object Names (page 192)

- Queue Commands

  CHGMQMQ, Change MQM Queue (page 122)
  CLRMQMQ, Clear MQM Queue (page 133)
  CPYMQMQ, Copy MQM Queue (page 144)
  CRTMQMQ, Create MQM Queue (page 168)
  DLTMQMQ, Delete MQM Queue (page 185)
  DSPMQMQ, Display MQM Queue (page 195)
  WRKMQMMSG, Work with MQM Messages (page 235)
  WRKMQMQ, Work with MQM Queues (page 237)

- Queue Manager Commands

  CCTMQM, Connect Message Queue Manager (page 106)
  CHGMQM, Change Message Queue Manager (page 107)
  CRTMQM, Create Message Queue Manager (page 155)
  DLTMQM, Delete Message Queue Manager (page 182)
  DSCMQM, Disconnect Message Queue Manager (page 186)
  DSPMQM, Display Message Queue Manager (page 187)
  ENDMQM, End Message Queue Manager (page 196)
  STRMQM, Start Message Queue Manager (page 218)

- Process Commands

  CHGMQMPRC, Change MQM Process (page 120)
  CPYMQMPRC, Copy MQM Process (page 142)
  CRTMQMPRC, Create MQM Process (page 166)
  DLTMQMPRC, Delete MQM Process (page 184)
  DSPMQMPRC, Display MQM Process (page 194)
  WRKMQMPRC, Work with MQM Processes (page 236)

- Security Commands

  DSPMQMAUT, Display MQM Object Authority (page 188)
  GRTMQMAUT, Grant MQM Object Authority (page 200)
  RVKMQMAUT, Revoke MQM Object Authority (page 213)

- Trace Commands

  ENDMQMSRV, End MQM Service Job (page 199)
  STRMQMSRV, Start MQM Service Job (page 226)
  TRCMQM, Trace MQM Job (page 228)

Names

The names of the following MQSeries objects can have up to 48 single-byte characters:

- Queue manager
- Queues
- Process definitions

The names of channels are restricted to 20 single-byte characters.

The characters that can be used for all MQSeries 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)

Notes:

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.

System defaults

Before you can use the system default values for the attributes of MQSeries objects, that is:

- *SYSDFTCHL for channels
- *SYSDFTQ for queues
- *SYSDFTPRC for processes

You should establish a set of defaults suitable to your system.

The sample program `AMQSDEF4` creates the objects that are needed and assigns values to their attributes. The default objects are:

SYSTEM.DEFAULT.ALIAS.QUEUE
Default alias queue

SYSTEM.DEFAULT.LOCAL.QUEUE
Default local queue

SYSTEM.DEFAULT.REMOTE.QUEUE
Default remote queue

SYSTEM.DEFAULT.PROCESS
Default MQI process object

SYSTEM.DEF.SENDER
Default Sender Channel

SYSTEM.DEF.SERVER
Default Server Channel

SYSTEM.AUTO.RECEIVER
Default Automatically Created Receiver Channel

SYSTEM.DEF.RECEIVER
Default Receiver Channel

SYSTEM.DEF.REQUESTER
Default Requester Channel

SYSTEM.DEFAULT.MODEL.QUEUE
Default Model Queue

SYSTEM.AUTO.SVRCONN
Default Automatically Created Server Connection Channel

SYSTEM.DEF.SVRCONN
Default Server Connection Channel

SYSTEM.CHANNEL.INITQ
Default Channel Initiation Queue

SYSTEM.CHANNEL.SYNCQ
Default Channel Synchronization Queue

SYSTEM.ADMIN.PERFM.EVENT
Default Performance Related Event Queue

SYSTEM.ADMIN.QMGR.EVENT
Default Queue Manager Related Event Queue

SYSTEM.MQSC.REPLY.QUEUE
Default MQSC Reply Queue

SYSTEM.ADMIN.COMMAND.QUEUE
Default Administration Command Queue

SYSTEM.CICS.INITIATION.QUEUE
Default CICS for AS/400 Initiation Queue

SYSTEM.DEFAULT.INITIATION.QUEUE
Default Initiation Queue

SYSTEM.ADMIN.CHANNEL.EVENT
Default Channel Related Event Queue

SYSTEM.DEAD.LETTER.QUEUE
Default Undelivered Message Queue

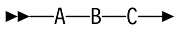
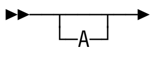
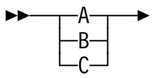
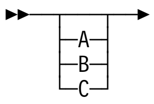
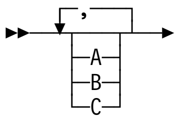
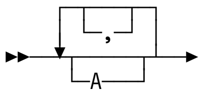
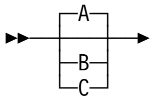
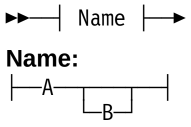
How to read syntax diagrams

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

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

Other conventions used in syntax diagrams are:

Table 6. How to read syntax diagrams

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

CCTMQM (Connect Message Queue Manager) Command

```
                                                Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──CCTMQM──────────────────────────────────────────────────────────────────────────────►◄
```

Purpose

The Connect Message Queue Manager (CCTMQM) command is used to connect to the Message Queue Manager before issuing other MQM commands.

The use of the CCTMQM command is optional, however, it can provide a significant performance improvement when issuing a sequence of two or more MQM commands from either the command line or from a CL program.

Note: This command has no effect on channel commands.

DSCMQM can subsequently be issued to disconnect the Message Queue Manager.

Required parameters

This command has no parameters.

CHGMQM (Change Message Queue Manager) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
                                               ┌─FORCE(*NO)──┐ ┌─TEXT(*SAME)──────────────┐
►►──CHGMQM──MQMNAME(queue-manager-name)─(P)────┼─────────────┼─┼──────────────────────────┼──────────►
                                               └─FORCE(*YES)─┘ └─TEXT(──┬─*BLANK──────┬─)─┘
                                                                        └─description─┘

  ┌─TRGITV(*SAME)──────────────┐ ┌─UDLMSGQ(*SAME)──────────────────────────────────────┐
►─┼────────────────────────────┼─┼─────────────────────────────────────────────────────┼──────────►
  └─TRIGITV(interval-value)────┘ └─UDLMSGQ(──┬─*NONE───────────────────────────────┬─)─┘
                                             └─undelivered-message-queue-name──────┘

  ┌─DFTTMQ(*SAME)──────────────────────────┐ ┌─MAXHDL(*SAME)──────────────────┐
►─┼────────────────────────────────────────┼─┼────────────────────────────────┼──────────────────►
  ├─(──────────────────────────────────────┤ └─MAXHDL(maximum-handle-limit)───┘
  └──┬─*NONE──────────────────────────────┬┘
     ├─default-transmission-queue-name────┤
     └─)──────────────────────────────────┘

  ┌─MAXUMSG(*SAME)──────────────────────────┐ ┌─AUTEVT(*SAME)─────┐ ┌─INHEVT(*SAME)─────┐
►─┼─────────────────────────────────────────┼─┼───────────────────┼─┼───────────────────┼────────►
  └─MAXUMSG(maximum-uncommitted-messages)───┘ └─AUTEVT(──┬─*NO──┬─)┘ └─INHEVT(──┬─*NO──┬─)┘
                                                         └─*YES─┘               └─*YES─┘

  ┌─LCLERREVT(*SAME)────┐ ┌─RMTERREVT(*SAME)────┐ ┌─PFREVT(*SAME)────┐ ┌─STRSTPEVT(*SAME)────┐
►─┼─────────────────────┼─┼─────────────────────┼─┼──────────────────┼─┼─────────────────────┼──►
  └─LCLERREVT(──┬─*NO──┬─)┘ └─RMTERREVT(──┬─*NO──┬─)┘ └─PRFEVT(──┬─*NO──┬─)┘ └─STRSTPEVT(──┬─*NO──┬─)┘
                └─*YES─┘                  └─*YES─┘             └─*YES─┘                  └─*YES─┘

  ┌─CHAD(*SAME)───────┐ ┌─CHADEV(*SAME)─────┐ ┌─CHADEXIT(*SAME)─────────────────────────────┐
►─┼───────────────────┼─┼───────────────────┼─┼─────────────────────────────────────────────┼──►◄
  └─CHAD(──┬─*NO──┬─)─┘ └─CHADEV(──┬─*NO──┬─)┘ └─CHADEXIT(──┬─*NONE───────────────────────┬─)─┘
           └─*YES─┘                └─*YES─┘                 └─library-name/program-name───┘
```

Note:
P All parameters preceding this point can be specified positionally.

Required parameters

MQMNAME

Specifies the name of the message queue manager.

For a remote request issued from the administration utility, this parameter specifies the name of the target message-queue manager on which the application is to be processed.

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

Optional parameters

FORCE

Specifies whether the command should be forced to complete if a new value is specified for DFTTMQ and an application has a remote queue open that would be affected by the changes.

***NO** The command fails if an open remote queue would be affected.

***YES** The command is forced to complete successfully.

TEXT

Specifies text that briefly describes the queue-manager definition.

***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).

TRGITV

Specifies the trigger time interval, expressed in milliseconds, to be used 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, 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.

<u>*SAME</u>: The attribute is unchanged.

interval-value: Specify a value in the range 0 through 99 999 999.

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.

<u>*SAME</u>: 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 the local queue that is to be used as the undelivered-message 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:

<u>*SAME</u>

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.

MAXHDL

Specifies the maximum number of handles that any one job can have open at the same time.

<u>*SAME</u>: The attribute is unchanged.

maximum-handle-limit: Specify a value in the range 0 through 999 999 999.

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 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.

***SAME** The attribute is unchanged.

maximum-uncommitted-messages Specify a value in the range 1 through 10 000. However, if you are using the MQSeries for AS/400 Administration Utility with a platform other than AS/400, the limits are 1 through 999 999 999.

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.

INHEVT
Specifies whether inhibit (Inhibit Get and Inhibit Put) events are generated.

The possible values are:

***SAME**

The attribute is unchanged.

***NO**

Inhibit events are not generated.

***YES**

Inhibit events are generated.

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.

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.

PFREVT
Specifies whether performance-related events are generated.

The possible values are:

***SAME**

The attribute is unchanged.

***NO**

Performance-related events are not generated.

***YES**

Performance-related events are generated.

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.

CHAD
Specifies whether receiver and server-connection channels can be 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.

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.

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.

| *library-name/program-name*

| You must specify a fully-qualified name, when you specify an automatic
| channel-definition user exit. In this instance, the libraries defined as *LIBL and
| *CURLIB are not permitted.

CHGMQMCHL (Change MQM Channel) Command

```
                                                          Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                        ┌─TRPTYPE(*SAME)──────┐  ┌─TEXT(*SAME)────────────────┐
►►──CHGMQMCHL──CHLNAME(channel-name)(P)─┼─────────────────────┼──┼────────────────────────────┼──────────►
                                        └─TRPTYPE(─┬─*LU62─┬─)┘  └─TEXT(─┬─*BLANK──────┬─)──┘
                                                   └─*TCP──┘             └─description─┘

   ┌─CONNAME(*SAME)─────────────────────┐ ┌─TMQNAME(*SAME)───────────────────────────┐
►──┼────────────────────────────────────┼─┼──────────────────────────────────────────┼────────────────────►
   └─CONNAME(─┬─*NONE────────────┬─(1)─)┘ └─TMQNAME(transmission-queue-name)─(2)─────┘
              └─connection-name──┘

   ┌─MCANAME(*SAME)────────────────────────────────┐ ┌─MCAUSERID(*SAME)─────────────────────┐ ┌─BATCHSIZE(*SAME)──────────┐
►──┼───────────────────────────────────────────────┼─┼──────────────────────────────────────┼─┼───────────────────────────┼──►
   └─MCANAME(─┬─*NONE──────────────────────┬─(1)─)─┘ └─MCAUSERID(─┬─*NONE────────────────┬─)┘ └─BATCHSIZE(batch-size)(4)─┘
              └─library-name/program-name──┘                      ├─*PUBLIC──────────────┤
                                                                  └─mca-user-identifier──┘

   ┌─DSCITV(*SAME)──────────────────────┐ ┌─SHORTTMR(*SAME)────────────────────────┐ ┌─SHORTRTY(*SAME)──────────────────┐
►──┼────────────────────────────────────┼─┼────────────────────────────────────────┼─┼──────────────────────────────────┼──►
   └─DSCITV(disconnect-interval)─(2)────┘ └─SHORTTMR(short-retry-interval)─(2)─────┘ └─SHORTRTY(short-retry-count)─(2)──┘

   ┌─LONGTMR(*SAME)───────────────────────┐ ┌─LONGRTY(*SAME)──────────────────┐ ┌─SCYEXIT(*SAME)─────────────────────────────┐
►──┼──────────────────────────────────────┼─┼─────────────────────────────────┼─┼────────────────────────────────────────────┼──►
   └─LONGTMR(long-retry-interval)─(2)─────┘ └─LONGRTY(long-retry-count)─(2)───┘ └─SCYEXIT(─┬─*NONE──────────────────────┬─)─┘
                                                                                           └─library-name/program-name──┘

   ┌─SCYUSRDATA(*SAME)─────────────────────────┐ ┌─SNDEXIT(*SAME)─────────────────────────────┐
►──┼───────────────────────────────────────────┼─┼────────────────────────────────────────────┼────────────────►
   └─SCYUSRDATA(─┬─*NONE─────────────────────┬─)┘ └─SNDEXIT(─┬─*NONE──────────────────────────┬─)┘
                 └─security-exit-user-data───┘              │ ┌─,─────────────────────────┐ │
                                                            └─▼─library-name/program-name─┴─┘

   ┌─SNDUSRDATA(*SAME)────────────────────────┐ ┌─RCVEXIT(*SAME)────────────────────────────────────────────┐
►──┼──────────────────────────────────────────┼─┤                                                           ├──►
   └─SNDUSRDATA(─┬─*NONE───────────────────┬─)┘ │          ┌─,────────────────────────────────────┐         │
                 │ ┌─,──────────────────┐  │    └─RCVEXIT(─▼─*NONE──library-name/program-name─────┴──────)──┘
                 └─▼─send-exit-user-data┴──┘

   ┌─RCVUSRDATA(*SAME)──────────────────────────────┐ ┌─MSGEXIT(*SAME)───────────────────────────────────┐
►──┼────────────────────────────────────────────────┼─┼──────────────────────────────────────────────────┼──►
   └─RCVUSRDATA(─┬─*NONE─────────────────────────┬─)┘ └─MSGEXIT(─┬─*NONE──────────────────────────────┬─)┘
                 │ ┌─,───────────────────────┐   │               │ ┌─,───────────────────────────┐    │
                 └─▼─receive-exit-user-data──┴───┘               └─▼─library-name/program-name───┴(4)─┘

   ┌─MSGUSRDATA(*SAME)──────────────────────────────────┐ ┌─MSGRTYEXIT(*SAME)────────────────────────────────┐
►──┼────────────────────────────────────────────────────┼─┼──────────────────────────────────────────────────┼──►
   └─MSGUSRDATA(─┬─*NONE─────────────────────────────┬─)┘ └─MSGRTYEXIT(─┬─*NONE──────────────────────┬─)(3)─┘
                 │ ┌─,───────────────────────┐       │                  └─library-name/program-name──┘
                 └─▼─message-exit-user-data──┴─(4)───┘

   ┌─MSGRTYDATA(*SAME)──────────────────────────────────┐ ┌─MSGRTYNBR(*SAME)───────────────────────────┐
►──┼────────────────────────────────────────────────────┼─┼────────────────────────────────────────────┼────────►
   └─MSGRTYDATA(─┬─*NONE─────────────────────────┬─)(3)─┘ └─MSGRTYNBR(message-retry-number)─(3)───────┘
                 └─message-retry-exit-user-data──┘

   ┌─MSGRTYITV(*SAME)──────────────────────────────┐ ┌─PUTAUT(*SAME)─────────────┐ ┌─SEQNUMWRAP(*SAME)───────────────────────────────┐
►──┼───────────────────────────────────────────────┼─┼───────────────────────────┼─┼─────────────────────────────────────────────────┼──►
   └─MSGRTYITV(message-retry-interval)─(3)─────────┘ └─PUTAUT(─┬─*DFT─┬─)─(3)────┘ └─SEQNUMWRAP(sequence-number-wrap-value)─(4)──────┘
                                                               └─*CTX─┘

   ┌─MAXMSGLEN(*SAME)────────────────────────┐ ┌─CVTMSG(*SAME)─────────────┐ ┌─HRTBTINTVL(*SAME)──────────────────┐
►──┼─────────────────────────────────────────┼─┼───────────────────────────┼─┼────────────────────────────────────┼──────►
   └─MAXMSGLEN(maximum-message-length)───────┘ └─CVTMSG(─┬─*YES─┬─)─(2)────┘ └─HRTBTINTVL(heart-beat-interval)────┘
                                                         └─*NO──┘

   ┌─NPMSPEED(*SAME)─────────────────┐
►──┼─────────────────────────────────┼─────────────────────────────────────────────────────────────────────────►◄
   └─NPMSPEED(─┬─*FAST───┬─)─(4)─────┘
               └─*NORMAL─┘
```

Notes:

1 Valid only for sender, server or requester channels.
2 Valid only for sender or server channels.
3 Valid only for receiver or requester channels.
4 Valid only for sender, server, receiver or requester channels.
P All parameters preceding this point can be specified positionally.

Purpose

The Change MQM Channel (CHGMQMCHL) command changes the specified attributes of an existing channel definition.

Restriction

To use this command, you must be signed on as QPGMR, or QSYSOPR, or have *ALLOBJ authority.

Required parameters

CHLNAME
Specifies the name of the channel definition.

Optional parameters

TRPTYPE
Specifies the transmission protocol.

*SAME: The attribute is unchanged.

***LU62**: SNA LU 6.2.

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

TEXT
Specifies text that briefly describes the channel definition.

*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).

CONNAME
Specifies the name of the machine to be connected.

*SAME: The attribute is unchanged.

NONE: The connection name is blank.

connection-name: Specify the name of the machine as required for 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.

A connection name is required if the channel type (CHLTYPE) is *RQSTR or *SDR.

TMQNAME
Specifies the name of the transmission queue.

*SAME: The attribute is unchanged.

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. For other channel types this parameter must not be specified.

MCANAME
This parameter is reserved and should not be used.

***SAME**: The attribute is unchanged.

***NONE**: The MCA program name is blank.

| This parameter cannot be specified for a channel type (CHLTYPE) of *RCVR or
| *SVRCN.

MCAUSERID
Message channel agent user identifier. 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.

BATCHSIZE
Specifies the maximum number of messages that should be sent down a channel before a checkpoint is taken.

| This parameter cannot be specified for a channel type (CHLTYPE) of *SVRCN.

***SAME**: The attribute is unchanged.

batch-size: Specify a value in the range 1 through 9999.

DSCITV
Specifies the disconnect interval for a sender or server (*SDR or *SVR) channel. This defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before closing the channel.

***SAME**: The attribute is unchanged.

disconnect-interval: Specify a value in the range 0 through 999 999. The value zero means an indefinite wait.

SHORTTMR
Specifies the short retry wait interval for a sender or server (*SDR or *SVR) channel that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine.

***SAME**: The attribute is unchanged.

short-retry-interval: Specify a value in the range 0 through 999 999 999.

SHORTRTY
Specifies the short retry count for a sender or server (*SDR or *SVR) channel 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.

<u>*SAME</u>: The attribute is unchanged.

short-retry-count: Specify a value in the range 0 through 999 999 999. A value of zero means that no retries are allowed.

LONGTMR
Specifies the long retry wait interval for a sender or server (*SDR or *SVR) channel that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by SHORTRTY has been exhausted.

<u>*SAME</u>: The attribute is unchanged.

long-retry-interval: Specify a value in the range 1 through 999 999 999.

LONGRTY
Specifies the long retry count for a sender or server (*SDR or *SVR) channel 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.

<u>*SAME</u>: The attribute is unchanged.

long-retry-count: Specify a value in the range 0 through 999 999 999. A value of zero means that no retries are allowed.

SCYEXIT
Specifies the entry point 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.

<u>*SAME</u>: The attribute is unchanged.

***NONE**: The send exit program is not invoked.

library-name/program-name: Specify the fully-qualified name of the security-exit program.

SCYUSRDATA
Specifies a maximum of 32 characters of user data that is passed to the security-exit program.

<u>*SAME</u>: The attribute is unchanged.

***NONE**: The user data for the security exit is not specified.

security-exit-user-data: Specify the user data for the security 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.

A list of exit programs may not be altered. If you need to add or delete a program, you must redefine the list.

| You can specify the names of up to 10 exit programs by specifying multiple
| strings separated by commas.

<u>*SAME</u>: The attribute is unchanged.

***NONE**: The send-exit program is not invoked.

library-name/program-name: Specify the fully-qualified name of the send exit program.

SNDUSRDATA
Specifies a maximum of 32 characters of user data that is passed to the send exit program.

| You can specify up to 10 strings, each of length 32 characters. The first string
| of data is passed to the first message exit specified, the second string to the
| second exit, and so on.

<u>*SAME</u>: 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.

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.

A list of exit programs may not be altered. If you need to add or delete a program, you must redefine the list.

| You can specify the names of up to 10 exit programs by specifying multiple
| strings separated by commas.

<u>*SAME</u>: The attribute is unchanged.

***NONE**: The receive exit program is not invoked.

library-name/program-name: Specify the fully-qualified name for the receive exit.

RCVUSRDATA
Specifies a maximum of 32 characters of user data that is passed to the receive exit program.

| You can specify up to 10 strings, each of length 32 characters. The first string
| of data is passed to the first message exit specified, the second string to the
| second exit, and so on.

<u>*SAME</u>: The attribute is unchanged.

***NONE**: The user data for the receive exit is not specified.

receive-exit-user-data: Specify a maximum of 32 characters of user data for the receive 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.

| This parameter cannot be specified for a channel type (CHLTYPE) of *SVRCN.

A list of exit programs may not be altered. If you need to add or delete a program, you must redefine the list.

| You can specify the names of up to 10 exit programs by specifying multiple
| strings separated by commas.

<u>*SAME</u>: The attribute is unchanged.

***NONE**: The message exit program is not invoked.

library-name/program-name: Specify the fully-qualified name for the message exit program.

MSGUSRDATA

Specifies user data that is passed to the message exit program.

| You can specify up to 10 strings, each of length 32 characters. The first string
| of data is passed to the first message exit specified, the second string to the
| second exit, and so on.

| This parameter cannot be specified for a channel type (CHLTYPE) of *SVRCN.

<u>*SAME</u>: 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.

MSGRTYEXIT

Specifies the entry point of the program to be called as the message retry exit for a receiver or requester (*RCVR, *RQSTR) channel.

<u>*SAME</u> The attribute is unchanged.

***NONE** The message retry exit program is not invoked.

library-name/program-name: Specify the fully qualified name for the message retry exit program.

MSGRTYDATA

Specifies that user data is passed to the message retry exit program.

<u>*SAME</u> 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.

MSGRTYITV

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

| This is valid for a receiver or requester (*RCVR, *RQSTR) channel only.

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.

***SAME** The attribute is unchanged.

message-retry-number: Specify a value in the range 0 through 999 999 999. A value of zero signifies that the retry will be performed as soon as possible.

MSGRTYNBR
Specifies the number of times the channel retries before it concludes it cannot deliver the message.

| This is valid for a receiver or requester (*RCVR, *RQSTR) channel only.

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.

***SAME** The attribute is unchanged.

message-retry-number: Specify a value in the range 0 through 999 999 999. A value of zero signifies that no retries will be performed.

PUTAUT
Specifies whether the user identifier in the context information associated with a message should be used to establish authority to put the message on the destination queue. This applies only to receiver and requester (*RCVR and *RQSTR) channels.

***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. Context authority is used.

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.

This parameter cannot be specified for a channel type (CHLTYPE) of *SVRCN.

***SAME**: The attribute is unchanged.

sequence-number-wrap-value: Specify a value in the range 100 through 999 999 999.

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.

***SAME**: The attribute is unchanged.

maximum-message-length: Specify a value in the range 0 through 4 194 304. A value of zero signifies that the maximum length is unlimited.

CVTMSG

Specifies whether the application data in the message should be converted before the message is transmitted. This applies only to sender and server (*SDR and *SVR) channels.

<u>*SAME</u> 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.

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:

<u>*SAME</u> The attribute is unchanged.

heart-beat-interval: Specify a value in the range 0 through 999 999 999. A value of zero signifies that no retries will be performed.

Note, that for implementation reasons, the maximum heartbeat interval that can be used is 999 999. Values exceeding this are treated as 999 999. A value of zero means that no heartbeat exchanges are to take place.

NPMSPEED

Specifies whether the channel supports fast-nonpersistent messages for sender, server, receiver, or requester (*SDR, *SVR, *RCVR, *RQSTR) channels.

The possible values are:

<u>*SAME</u> The attribute is unchanged.

***FAST** The channel supports fast nonpersistent messages.

***NORMAL** The channel does not support fast nonpersistent messages.

CHGMQMPRC (Change MQM Process) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
                                          ┌─TEXT(*SAME)──────────────┐ ┌─APPTYPE(*SAME)──────────┐
►►──CHGMQMPRC──PRCNAME(process-name)─(P)──┼──────────────────────────┼─┼─────────────────────────┼──────────────►
                                          └─TEXT(──┬─*BLANK──────┬─)─┘ └─APPTYPE(──┬─*OS400─────┬─)─┘
                                                   └─description─┘                 ├─*CICS──────┤
                                                                                   └─user-value─┘

   ┌─APPID(*SAME)──────────────┐  ┌─USRDATA(*SAME)───────────┐  ┌─ENVDATA(*SAME)─────────────────────┐
►──┼───────────────────────────┼──┼──────────────────────────┼──┼────────────────────────────────────┼────────►◄
   └─APPID(application-id)─────┘  └─USRDATA(──┬─*NONE─────┬─)─┘  └─ENVDATA(──┬─*NONE────────────┬─)─┘
                                              └─user-data─┘                 └─environment-data─┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Change MQM Process (CHGMQMPRC) command changes the specified attributes of an existing MQM process definition.

Required parameters

PRCNAME

The name of the process definition to be changed.

process-name: Specify the name of the process definition. The maximum length of the string is 48 bytes. See "Names" on page 102.

Optional parameters

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).

<u>*SAME</u>: The attribute is unchanged.

***BLANK**: The text is set to a blank string.

description: Specify no more than 64 characters enclosed in apostrophes.

APPTYPE

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

<u>*SAME</u>: The attribute is unchanged.

***OS400**: Represents an OS/400 application.

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

user-value: User-defined application type in the range 65 536 through 999 999 999.

APPID

Application identifier. This is the name of the application to be started, on the platform for which the command is processing, and is typically a program name and library name.

***SAME**: The attribute is unchanged.

application-id: The maximum length is 256 characters.

USRDATA

A character string that contains user information pertaining to the application, as defined by APPID, to be started.

***SAME**: The attribute is unchanged.

***NONE**: The user data is blank.

user-data: Specify up to 128 characters of user data.

ENVDATA

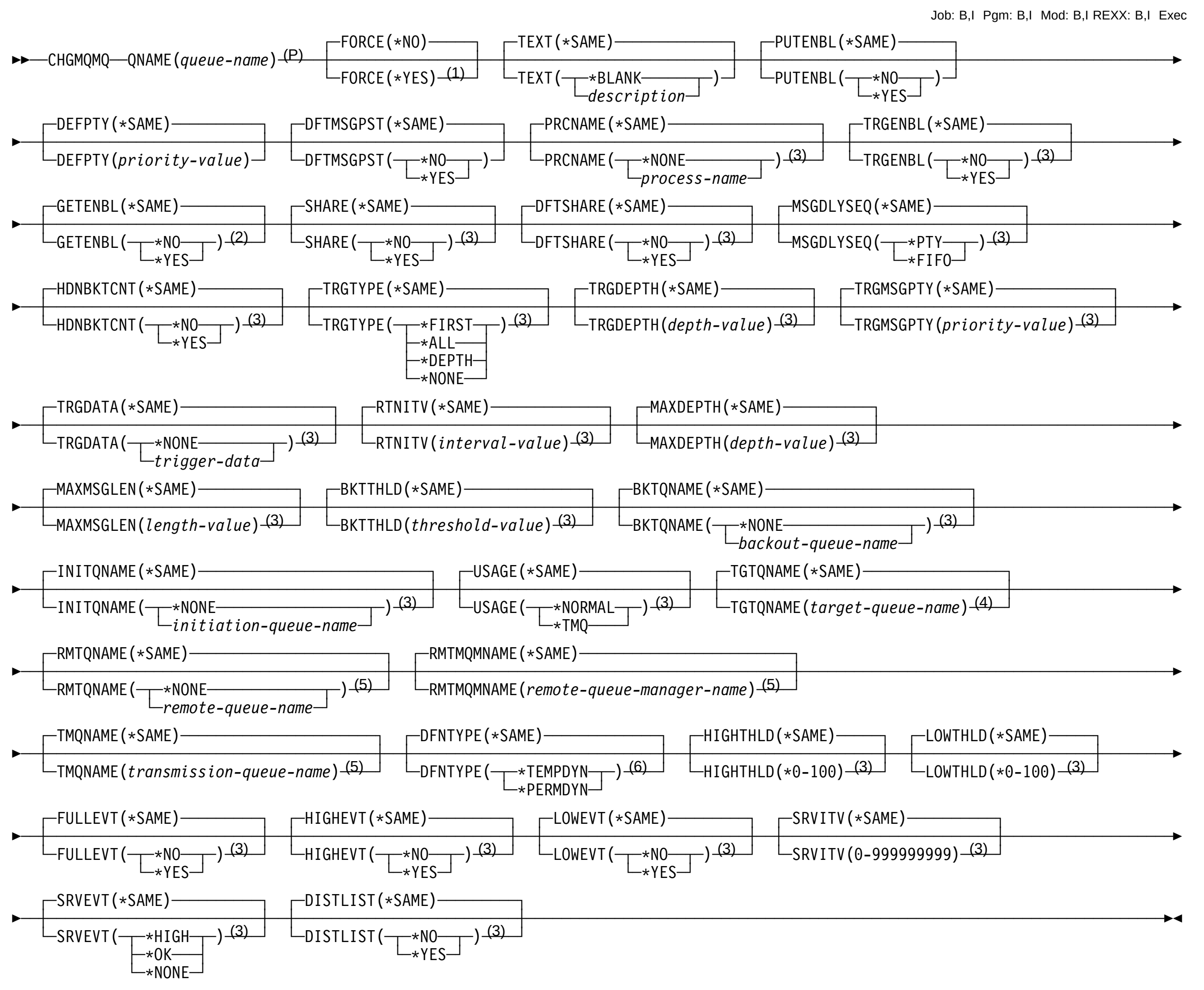
A character string that contains environment information pertaining to the application, as defined by APPID, to be started.

***SAME**: The attribute is unchanged.

***NONE**: The environment data is blank.

environment-data: The maximum length is 128 characters.

CHGMQMQ (Change MQM Queue) Command


Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec
CHGMQMQ QNAME(queue-name) (P) FORCE(*NO) FORCE(*YES) (1) TEXT(*SAME) TEXT(*BLANK description) PUTENBL(*SAME) PUTENBL(*NO *YES)
DEFPTY(*SAME) DEFPTY(priority-value) DFTMSGPST(*SAME) DFTMSGPST(*NO *YES) PRCNAME(*SAME) PRCNAME(*NONE process-name) (3) TRGENBL(*SAME) TRGENBL(*NO *YES) (3)
GETENBL(*SAME) GETENBL(*NO *YES) (2) SHARE(*SAME) SHARE(*NO *YES) (3) DFTSHARE(*SAME) DFTSHARE(*NO *YES) (3) MSGDLYSEQ(*SAME) MSGDLYSEQ(*PTY *FIFO) (3)
HDNBKTCNT(*SAME) HDNBKTCNT(*NO *YES) (3) TRGTYPE(*SAME) TRGTYPE(*FIRST *ALL *DEPTH *NONE) (3) TRGDEPTH(*SAME) TRGDEPTH(depth-value) (3) TRGMSGPTY(*SAME) TRGMSGPTY(priority-value) (3)
TRGDATA(*SAME) TRGDATA(*NONE trigger-data) (3) RTNITV(*SAME) RTNITV(interval-value) (3) MAXDEPTH(*SAME) MAXDEPTH(depth-value) (3)
MAXMSGLEN(*SAME) MAXMSGLEN(length-value) (3) BKTTHLD(*SAME) BKTTHLD(threshold-value) (3) BKTQNAME(*SAME) BKTQNAME(*NONE backout-queue-name) (3)
INITQNAME(*SAME) INITQNAME(*NONE initiation-queue-name) (3) USAGE(*SAME) USAGE(*NORMAL *TMQ) (3) TGTQNAME(*SAME) TGTQNAME(target-queue-name) (4)
RMTQNAME(*SAME) RMTQNAME(*NONE remote-queue-name) (5) RMTMQMNAME(*SAME) RMTMQMNAME(remote-queue-manager-name) (5)
TMQNAME(*SAME) TMQNAME(transmission-queue-name) (5) DFNTYPE(*SAME) DFNTYPE(*TEMPDYN *PERMDYN) (6) HIGHTHLD(*SAME) HIGHTHLD(*0-100) (3) LOWTHLD(*SAME) LOWTHLD(*0-100) (3)
FULLEVT(*SAME) FULLEVT(*NO *YES) (3) HIGHEVT(*SAME) HIGHEVT(*NO *YES) (3) LOWEVT(*SAME) LOWEVT(*NO *YES) (3) SRVITV(*SAME) SRVITV(0-999999999) (3)
SRVEVT(*SAME) SRVEVT(*HIGH *OK *NONE) (3) DISTLIST(*SAME) DISTLIST(*NO *YES) (3)

Notes:

1 Valid only for local, remote, and alias queues.
2 Valid only for local, model, and alias queues.
3 Valid only for local and model queues.
4 Valid only for alias queues.
5 Valid only for remote queues.
6 Valid only for model queues.
P All parameters preceding this point can be specified positionally.

Purpose

The Change MQM Queue (CHGMQMQ) command changes the attributes of an existing MQM queue.

Required parameters

QNAME

The name of the queue to be changed.

queue-name: Specify the name of the queue.

Optional parameters

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 has a remote queue open that 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 resolved 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.

<u>*NO</u>: The command fails if the relevant conditions are true.

***YES**: The command is forced to complete successfully even if the relevant conditions are true.

This parameter cannot be specified for a model (*MDL) queue.

TEXT

Specifies text that briefly describes the queue definition.

<u>*SAME</u>: 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).

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.

<u>***SAME**</u>: The attribute is unchanged.

***NO**: Messages cannot be added to the queue.

***YES**: Messages can be added to the queue by authorized applications.

DFTPTY
Specifies the default priority of messages put on the queue.

<u>***SAME**</u>: The attribute is unchanged.

priority-value: Specify the default priority for the queue. The value must be in the range zero through the maximum priority supported, that is, (9).

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.

<u>***SAME**</u>: 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.

PRCNAME
Specifies the local name of the MQM process, for a local or model (*LCL or *MDL) queue, 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.

<u>***SAME**</u>: The attribute is unchanged.

***NONE**: The process name is blank.

process-name: Specify the name of the MQM process.

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.

<u>***SAME**</u>: The attribute is unchanged.

***NO**: Do not write trigger messages to the initiation queue.

***YES**: Triggering is active; trigger messages are written to the initiation queue.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

<u>*SAME</u>: The attribute is unchanged.

***NO**: Applications cannot retrieve messages from the queue.

***YES**: Suitably authorized applications can retrieve messages from the queue.

| This parameter cannot be specified for a remote (*RMT) queue.

SHARE
Specifies whether multiple instances of applications, can open this queue for input,

<u>*SAME</u>: 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.

This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

DFTSHARE
Specifies the default share option for applications opening this queue for input.

<u>*SAME</u>: 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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

MSGDLYSEQ
Specifies the message delivery sequence.

<u>*SAME</u>: 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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

HDNBKTCNT
Specifies whether the count of backed out messages should be saved (hardened) across restarts of the message queue manager.

<u>*SAME</u>: The attribute is unchanged.

***NO**: The backout count is not hardened.

***YES**: The backout count is hardened.

Note: On MQSeries for AS/400 the count is *always* hardened, regardless of the setting of this attribute.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

***SAME**: The attribute is unchanged.

***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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

TRGDEPTH

Specifies, for TRIGTYPE(*DEPTH), the number of messages that initiates a trigger message to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

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

depth-value: Specify a value in the range 1 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

***SAME**: The attribute is unchanged.

priority-value: Specify a value in the range of priority values that are supported, that is, 0 through 9.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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 that is started by the monitor.

For a transmission queue, you can use this parameter to specify the name of the channel to be started.

Note: An application program can issue a call to MQSET to change the value of this attribute.

***SAME**: The attribute is unchanged.

***NONE**: No trigger data is specified.

trigger-data: Specify up to 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).

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

RTNITV
Specifies the retention interval. This interval is the number of hours for which the queue may 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 may be used to determine when a queue is no longer required.

Note: The message queue manager does not delete queues nor does it prevent queues from being deleted if their retention interval has not expired. It is the user's responsibility to take any required action.

***SAME**: The attribute is unchanged.

interval-value: Specify a value in the range 0 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

MAXDEPTH
Specifies the maximum number of messages allowed on the queue. However, other factors may 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.

***SAME**: The attribute is unchanged.

depth-value: Specify a value in the range 0 through 640 000. However, if you are using the MQSeries for AS/400 Administration Utility with a platform other than AS/400, the limits are 0 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

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

***SAME**: The attribute is unchanged.

length-value: Specify a value in the range 0 through 4 194 304.

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

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

BKTTHLD

Specifies the backout threshold. Apart from allowing this attribute to be queried, MQSeries for AS/400 takes no action based on the value of the attribute.

***SAME**: The attribute is unchanged.

threshold-value: Specify a value in the range 0 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

BKTQNAME

Specifies the backout-queue name. Apart from allowing this attribute to be queried, MQSeries for AS/400 takes no action based on the value of the attribute.

***SAME**: The attribute is unchanged.

***NONE**: No backout queue is specified.

backout-queue-name: Specify the backout queue name.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

INITQNAME

Specifies the name of the initiation queue. That is, the local queue for trigger messages relating to this new, or changed, queue.

Note: The initiation queue must be on the same queue manager.

***SAME**: The attribute is unchanged.

***NONE**: No initiation queue is specified.

initiation-queue-name: Specify the initiation queue name.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

USAGE

Specifies whether the queue is for normal usage, or for transmitting messages to a remote message queue manager.

***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 the *MQSeries Intercommunication* book.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

TGTQNAME

Specifies the name of the local or remote target queue, for which this queue is an alias.

Note: The target queue does not need to exist at this time but it must exist when a process attempts to open the alias queue.

***SAME**: The attribute is unchanged.

target-queue-name: Specify the name of the target queue.

RMTQNAME

Specifies the name of the queue on the remote system. If this definition is used for a queue-manager alias definition, RMTQNAME must be blank when the open occurs (*NONE can be specified).

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.

***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.

| This parameter can be specified for a remote (*RMT) queue only.

RMTMQMNAME

Specifies the name of the queue manager on the remote system.

If an application opens the local definition of a remote queue, RMTMQMNAME must not be the name of the local 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 local 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.

***SAME**: The attribute is unchanged.

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

Note: The name is not checked to ensure that it contains only those characters normally allowed for queue manager names.

| This parameter can be specified for a remote (*RMT) queue only.

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 instead 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 local queue manager.

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

***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.

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 in the range 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

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 in the range 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

LOWEVT
Specifies whether Queue Depth Low events are generated.

The possible values are:

<u>*SAME</u>

The attribute is unchanged.

***NO**

Queue Depth Low events are not generated.

***YES**

Queue Depth Low events are generated.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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:

<u>*SAME</u>

The attribute is unchanged.

interval-value

Specify a value in the range 0 through 999 999 999. The value is in units of milliseconds.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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 at least the time indicated by the SRVITV parameter.

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:

<u>*SAME</u>

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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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 applies to a model queue definition only.

The possible values are:

<u>*SAME</u>

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.

DISTLIST
Specifies whether or not the queue supports distribution lists

The possible values are:

<u>*SAME</u>

The attribute is unchanged.

***NO**

Distribution lists are not supported.

***YES**

Distribution lists are supported.

This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

CLRMQMQ (Clear MQM Queue) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
►►──CLRMQMQ──QNAME(queue-name)─(P)─────────────────────────────────────────────►◄
```

Notes:

1 This command is valid only for QTYPE *LCL.

P All parameters preceding this point can be specified positionally.

Purpose

The Clear MQM 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.

Required parameters

QNAME

The name of the local queue to be cleared.

queue-name: Specify the name of the queue.

CPYMQMCHL (Copy MQM Channel) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
                                                       ┌─REPLACE(*NO)──┐ ┌─TRPTYPE(*SAME)──────────┐
►►──CPYMQMCHL──FROMCHL(from-channel-name)──TOCHL(to-channel-name)(P)──┴─REPLACE(*YES)─┴─┴─TRPTYPE(─┬─*LU62─┬─)─┴──►
                                                                                       └─*TCP──┘

  ┌─TEXT(*SAME)──────────────┐ ┌─CONNAME(*SAME)──────────────────┐ ┌─TMQNAME(*SAME)──────────────────────────┐
►─┴─TEXT(─┬─*BLANK──────┬─)─┴─┴─CONNAME(─┬─*NONE───────────┬─(1)─)─┴─┴─TMQNAME(transmission-queue-name)(2)─┴──►
          └─description─┘                └─connection-name─┘

  ┌─MCANAME(*SAME)──────────────────────────────┐ ┌─MCAUSERID(*SAME)─────────────────────┐ ┌─BATCHSIZE(*SAME)────────┐
►─┴─MCANAME(─┬─*NONE─────────────────────┬─(1)─)─┴─┴─MCAUSERID(─┬─*NONE──────────────┬─)─┴─┴─BATCHSIZE(batch-size)(4)─┴──►
             └─library-name/program-name─┘                      ├─*PUBLIC────────────┤
                                                                └─mca-user-identifier─┘

  ┌─DSCITV(*SAME)──────────────────────┐ ┌─SHORTTMR(*SAME)────────────────────────┐ ┌─SHORTRTY(*SAME)─────────────────────┐
►─┴─DSCITV(disconnect-interval)(2)─┴─┴─SHORTTMR(short-retry-interval)(2)─┴─┴─SHORTRTY(short-retry-count)(2)─┴──►

  ┌─LONGTMR(*SAME)─────────────────────┐ ┌─LONGRTY(*SAME)───────────────────┐ ┌─SCYEXIT(*SAME)────────────────────────────┐
►─┴─LONGTMR(long-retry-interval)(2)─┴─┴─LONGRTY(long-retry-count)(2)─┴─┴─SCYEXIT(─┬─*NONE─────────────────────┬─)─┴──►
                                                                                   └─library-name/program-name─┘

  ┌─SCYUSRDATA(*SAME)───────────────────────────┐ ┌─SNDEXIT(*SAME)─────────────────────────┐
►─┴─SCYUSRDATA(─┬─*NONE───────────────────┬─)─┴─┴─SNDEXIT(─┬─*NONE─────────────────────┬─)─┴──►
                └─security-exit-user-data─┘                │ ┌─,───────────────────────┐ │
                                                           └─▼─library-name/program-name─┴─┘

  ┌─SNDUSRDATA(*SAME)───────────────────┐ ┌─RCVEXIT(*SAME)──────────────────────────────────────┐
►─┴─SNDUSRDATA(─┬─*NONE─────────────┬─)─┴─┤      ┌─,───────────────────────────────────┐        │
                │ ┌─,─────────────┐ │     └─RCVEXIT(─▼─*NONE──library-name/program-name─┴─)──────┘
                └─▼─send-exit-user-data─┴─┘

  ┌─RCVUSRDATA(*SAME)──────────────────────┐ ┌─MSGEXIT(*SAME)────────────────────────────┐
►─┴─RCVUSRDATA(─┬─*NONE────────────────┬─)─┴─┴─MSGEXIT(─┬─*NONE─────────────────────────┬─)─┴──►
                │ ┌─,────────────────┐ │                │ ┌─,─────────────────────────┐ │
                └─▼─receive-exit-user-data─┴─┘          └─▼─library-name/program-name─┴(4)─┘

  ┌─MSGUSRDATA(*SAME)──────────────────────────┐ ┌─MSGRTYEXIT(*SAME)───────────────────────────────┐
►─┴─MSGUSRDATA(─┬─*NONE────────────────────┬─)─┴─┴─MSGRTYEXIT(─┬─*NONE─────────────────────┬─)(3)─┴──►
                │ ┌─,──────────────────┐   │                   └─library-name/program-name─┘
                └─▼─message-exit-user-data─┴(4)─┘

  ┌─MSGRTYDATA(*SAME)───────────────────────────────────┐ ┌─MSGRTYNBR(*SAME)─────────────────────────┐
►─┴─MSGRTYDATA(─┬─*NONE─────────────────────────┬─)(3)─┴─┴─MSGRTYNBR(message-retry-number)(3)─┴──►
                └─message-retry-exit-user-data─┘

  ┌─MSGRTYITV(*SAME)──────────────────────────┐ ┌─PUTAUT(*SAME)──────────┐ ┌─SEQNUMWRAP(*SAME)───────────────────────────────┐
►─┴─MSGRTYITV(message-retry-interval)(3)─┴─┴─PUTAUT(─┬─*DFT─┬─)(3)─┴─┴─SEQNUMWRAP(sequence-number-wrap-value)(4)─┴──►
                                                        └─*CTX─┘

  ┌─MAXMSGLEN(*SAME)────────────────────────┐ ┌─CVTMSG(*SAME)──────────┐ ┌─HRTBTINTVL(*SAME)──────────────────┐
►─┴─MAXMSGLEN(maximum-message-length)─┴─┴─CVTMSG(─┬─*YES─┬─)(2)─┴─┴─HRTBTINTVL(heart-beat-interval)─┴──►
                                                   └─*NO──┘

  ┌─NPMSPEED(*SAME)─────────────────┐
►─┴─NPMSPEED(─┬─*FAST───┬─)(4)─┴──────────────────────────────────────────────────────────────────►◄
              └─*NORMAL─┘
```

Notes:

1 Valid only for sender, server or requester channels.

2 Valid only for sender or server channels.

3 Valid only for receiver or requester channels.

4 Valid only for receiver or requester channels.

P All parameters preceding this point can be specified positionally.

Purpose

The Copy MQM Channel (CPYMQMCHL) command creates a new channel definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing channel definition.

Restriction

To use this command, you must be signed on as QPGMR, or QSYSOPR, or have *ALLOBJ authority.

Required parameters

FROMCHL
Specifies the name of the existing channel definition that contains values for the attributes that are not specified in this command.

from-channel-name: Specify the name of the source MQM 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.

to-channel-name: Specify the name of the MQM channel being created.

Optional parameters

REPLACE
Specifies whether the new channel definition should replace an existing channel definition that has the same name.

<u>***NO**</u>: 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.

TRPTYPE
Specifies the transmission protocol.

<u>***SAME**</u>: The attribute is unchanged.

***LU62**: SNA LU 6.2.

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

TEXT
Specifies text that briefly describes the channel definition.

<u>***SAME**</u>: 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).

CONNAME
Specifies the name of the machine to be connected.

<u>***SAME**</u>: The attribute is unchanged.

NONE: The connection name is blank.

connection-name: Specify the name of the machine as required for 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.

A connection name is required if the channel type (CHLTYPE) is *RQSTR or *SDR.

TMQNAME
Specifies the name of the transmission queue.

*SAME: The attribute is unchanged.

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. For other channel types this parameter must not be specified.

MCANAME
This parameter is reserved and should not be used.

*SAME: The attribute is unchanged.

***NONE**: The MCA program name is blank.

| This parameter cannot be specified for a channel type (CHLTYPE) of *RCVR or
| *SVRCN.

MCAUSERID
Message channel agent user identifier. 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.

BATCHSIZE
Specifies the maximum number of messages that should be sent down a channel before a checkpoint is taken.

| This parameter cannot be specified for a channel type (CHLTYPE) of *SVRCN.

*SAME: The attribute is unchanged.

batch-size: Specify a value in the range 1 through 9999.

DSCITV
Specifies the disconnect interval for a sender or server (*SDR or *SVR) channel. This defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before closing the channel.

*SAME: The attribute is unchanged.

disconnect-interval: Specify a value in the range 0 through 999 999. The value zero means an indefinite wait.

SHORTTMR
Specifies the short retry wait interval for a sender or server (*SDR or *SVR) channel that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine.

<u>*SAME</u>: The attribute is unchanged.

short-retry-interval: Specify a value in the range 0 through 999 999 999.

SHORTRTY
Specifies the short retry count for a sender or server (*SDR or *SVR) channel 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.

<u>*SAME</u>: The attribute is unchanged.

short-retry-count: Specify a value in the range 0 through 999 999 999. A value of zero means that no retries are allowed.

LONGTMR
Specifies the long retry wait interval for a sender or server (*SDR or *SVR) channel that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by SHORTRTY has been exhausted.

<u>*SAME</u>: The attribute is unchanged.

long-retry-interval: Specify a value in the range 1 through 999 999 999.

LONGRTY
Specifies the long retry count for a sender or server (*SDR or *SVR) channel 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.

<u>*SAME</u>: The attribute is unchanged.

long-retry-count: Specify a value in the range 0 through 999 999 999. A value of zero means that no retries are allowed.

SCYEXIT
Specifies the entry point 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.

*SAME: The attribute is unchanged.

***NONE**: The send exit program is not invoked.

library-name/program-name: Specify the fully-qualified name of the security-exit program.

SCYUSRDATA
Specifies a maximum of 32 characters of user data that is passed to the security-exit program.

*SAME: The attribute is unchanged.

***NONE**: The user data for the security exit is not specified.

security-exit-user-data: Specify the user data for the security 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.

A list of exit programs may not be altered. If you need to add or delete a program, you must redefine the list.

| You can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

*SAME: The attribute is unchanged.

***NONE**: The send-exit program is not invoked.

library-name/program-name: Specify the fully-qualified name of the send exit program.

SNDUSRDATA
Specifies a maximum of 32 characters of user data that is passed to the send exit program.

| You can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first message exit specified, the second string to the second exit, and so on.

*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.

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.

A list of exit programs may not be altered. If you need to add or delete a program, you must redefine the list.

| You can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

*SAME: The attribute is unchanged.

***NONE**: The receive exit program is not invoked.

library-name/program-name: Specify the fully-qualified name for the receive exit.

RCVUSRDATA

Specifies a maximum of 32 characters of user data that is passed to the receive exit program.

| You can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first message exit specified, the second string to the second exit, and so on.

***SAME**: The attribute is unchanged.

***NONE**: The user data for the receive exit is not specified.

receive-exit-user-data: Specify a maximum of 32 characters of user data for the receive 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.

| This parameter cannot be specified for a channel type (CHLTYPE) of *SVRCN.

A list of exit programs may not be altered. If you need to add or delete a program, you must redefine the list.

| You can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

***SAME**: The attribute is unchanged.

***NONE**: The message exit program is not invoked.

library-name/program-name: Specify the fully-qualified name for the message exit program.

MSGUSRDATA

Specifies user data that is passed to the message exit program.

| You can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first message exit specified, the second string to the second exit, and so on.

| This parameter cannot be specified for a channel type (CHLTYPE) of *SVRCN.

***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.

MSGRTYEXIT

Specifies the entry point of the program to be called as the message retry exit for a receiver or requester (*RCVR, *RQSTR) channel.

***SAME** The attribute is unchanged.

***NONE** The message retry exit program is not invoked.

library-name/program-name: Specify the fully qualified name for the message retry exit program.

MSGRTYDATA
Specifies that user data is passed to the message retry exit program.

***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.

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

| This is valid for a receiver or requester (*RCVR, *RQSTR) channel only

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.

***SAME** The attribute is unchanged.

message-retry-number: Specify a value in the range 0 through 999 999 999. A value of zero signifies that the retry will be performed as soon as possible.

MSGRTYNBR
Specifies the number of times the channel retries before it concludes it cannot deliver the message.

| This is valid for a receiver or requester (*RCVR, *RQSTR) channel only.

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.

***SAME** The attribute is unchanged.

message-retry-number: Specify a value in the range 0 through 999 999 999. A value of zero signifies that no retries will be performed.

PUTAUT
Specifies whether the user identifier in the context information associated with a message should be used to establish authority to put the message on the destination queue. This applies only to receiver and requester (*RCVR and *RQSTR) channels.

***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. Context authority is used.

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.

| This parameter cannot be specified for a channel type (CHLTYPE) of *SVRCN.

<u>*SAME</u>: The attribute is unchanged.

sequence-number-wrap-value: Specify a value in the range 100 through 999 999 999.

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.

<u>*SAME</u>: The attribute is unchanged.

maximum-message-length: Specify a value in the range 0 through 4 194 304. A value of zero signifies that the maximum length is unlimited.

CVTMSG
Specifies whether the application data in the message should be converted before the message is transmitted. This applies only to sender and server (*SDR and *SVR) channels.

<u>*SAME</u> 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.

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:

<u>*SAME</u> The attribute is unchanged.

heart-beat-interval: Specify a value in the range 0 through 999 999 999. A value of zero signifies that no retries will be performed.

Note, that for implementation reasons, the maximum heartbeat interval that can be used is 999 999. Values exceeding this are treated as 999 999. A value of zero means that no heartbeat exchanges are to take place.

NPMSPEED
Specifies whether the channel supports fast-nonpersistent messages for sender, server, receiver, or requester (*SDR, *SVR, *RCVR, *RQSTR) channels.

The possible values are:

<u>*SAME</u> The attribute is unchanged.

***FAST** The channel supports fast nonpersistent messages.

***NORMAL** The channel does not support fast nonpersistent messages.

CPYMQMPRC (Copy MQM Process) Command

```
                                                                          Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
                                                           ┌─REPLACE(*NO)──┐  ┌─TEXT(*SAME)──────────────┐
►►──CPYMQMPRC──FROMPRC(from-process-name)──TOPRC(to-process-name)(P)─┴─REPLACE(*YES)─┘  └─TEXT(─┬─*BLANK──────┬─)─┘──────────►
                                                                                          └─description─┘

   ┌─APPTYPE(*SAME)──────────────┐  ┌─APPID(*SAME)─────────────┐  ┌─USRDATA(*SAME)───────────┐  ┌─ENVDATA(*SAME)─────────────────────┐
►──┴─APPTYPE(─┬─*OS400─────┬─)─┘──┴─APPID(application-id)─┘──┴─USRDATA(─┬─*NONE─────┬─)─┘──┴─ENVDATA(─┬─*NONE────────────┬─)─┘──►◄
              ├─*CICS──────┤                                            └─user-data─┘                 └─environment-data─┘
              └─user-value─┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Copy MQM Process (CPYMQMPRC) command creates an MQM process definition of the same type and, for attributes not specified in the command, with the same attribute values as an existing process definition.

Required parameters

FROMPRC
Specifies the name of the existing process definition to provide values for the attributes not specified in this command.

from-process-name: Specify the name of the source MQM 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.

to-process-name: Specify the name of the MQM process being created.

Optional parameters

REPLACE
Specifies whether the new process definition should replace an existing process definition with the same name.

***NO**: Do not replace the existing process definition. 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
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).

***SAME**: The attribute is unchanged.

***BLANK**: The descriptive information is blank.

description: Specify no more than 64 characters enclosed in apostrophes.

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

***SAME**: The attribute is unchanged.

***OS400**: Represents an OS/400 application.

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

user-value: User-defined application type in the range 65 536-999 999 999.

APPID
Application identifier. This is the name of the application to be started, on the platform for which the command is processing, and is typically a program name and library name.

***SAME**: The attribute is unchanged.

application-id: The maximum length is 256 characters.

USRDATA
A character string that contains user information pertaining to the application, as defined by APPID, to be started.

***SAME**: The attribute is unchanged.

***NONE**: The user data is blank.

user-data: Specify up to 128 characters of user data.

ENVDATA
A character string that contains environment information pertaining to the application, as defined by APPID, to be started.

***SAME**: The attribute is unchanged.

***NONE**: The environment data is blank.

environment-data: The maximum length is 128 characters.

CPYMQMQ (Copy MQM Queue) Command

```
                                                                   Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                                         ┌─REPLACE(*NO)──┐ ┌─TEXT(*SAME)──────────────┐
►►──CPYMQMQ──FROMQ(from-queue-name)──TOQ(to-queue-name)(P)┴─REPLACE(*YES)─┘ └─TEXT(─┬─*BLANK──────┬─)─┘────────►
                                                                               └─description─┘

  ┌─PUTENBL(*SAME)────────┐ ┌─DEFPTY(*SAME)───────────┐ ┌─DFTMSGPST(*SAME)───────┐ ┌─PRCNAME(*SAME)───────────────┐
►─┴─PUTENBL(─┬─*NO──┬─)─┘ └─DEFPTY(priority-value)─┘ └─DFTMSGPST(─┬─*NO──┬─)─┘ └─PRCNAME(─┬─*NONE────────┬─)(2)─┘──►
             └─*YES─┘                                              └─*YES─┘               └─process-name─┘

  ┌─TRGENBL(*SAME)───────────┐ ┌─GETENBL(*SAME)───────────┐ ┌─SHARE(*SAME)───────────┐ ┌─DFTSHARE(*SAME)───────────┐
►─┴─TRGENBL(─┬─*NO──┬─)(2)─┘ └─GETENBL(─┬─*NO──┬─)(1)─┘ └─SHARE(─┬─*NO──┬─)(2)─┘ └─DFTSHARE(─┬─*NO──┬─)(2)─┘────►
             └─*YES─┘                   └─*YES─┘                 └─*YES─┘                     └─*YES─┘

  ┌─MSGDLYSEQ(*SAME)────────────┐ ┌─HDNBKTCNT(*SAME)───────────┐ ┌─TRGTYPE(*SAME)────────────┐ ┌─TRGDEPTH(*SAME)──────────────┐
►─┴─MSGDLYSEQ(─┬─*PTY──┬─)(2)─┘ └─HDNBKTCNT(─┬─*NO──┬─)(2)─┘ └─TRGTYPE(─┬─*FIRST─┬─)(2)─┘ └─TRGDEPTH(depth-value)(2)─┘─►
               └─*FIFO─┘                     └─*YES─┘                   ├─*ALL───┤
                                                                        ├─*DEPTH─┤
                                                                        └─*NONE──┘

  ┌─TRGMSGPTY(*SAME)────────────────┐ ┌─TRGDATA(*SAME)──────────────────┐ ┌─RTNITV(*SAME)─────────────────┐
►─┴─TRGMSGPTY(priority-value)(2)─┘ └─TRGDATA(─┬─*NONE────────┬─)(2)─┘ └─RTNITV(interval-value)(2)─┘──────────►
                                              └─trigger-data─┘

  ┌─MAXDEPTH(*SAME)──────────────┐ ┌─MAXMSGLEN(*SAME)────────────────┐ ┌─BKTTHLD(*SAME)──────────────────┐
►─┴─MAXDEPTH(depth-value)(2)─┘ └─MAXMSGLEN(length-value)(2)─┘ └─BKTTHLD(threshold-value)(2)─┘──────────►

  ┌─BKTQNAME(*SAME)─────────────────────────┐ ┌─INITQNAME(*SAME)─────────────────────────────┐ ┌─USAGE(*SAME)─────────────┐
►─┴─BKTQNAME(─┬─*NONE──────────────┬─)(2)─┘ └─INITQNAME(─┬─*NONE────────────────────┬─)(2)─┘ └─USAGE(─┬─*NORMAL─┬─)(2)─┘─►
              └─backout-queue-name─┘                     └─initiation-queue-name─┘                   └─*TMQ────┘

  ┌─TGTQNAME(*SAME)───────────────────────┐ ┌─RMTQNAME(*SAME)──────────────────────────┐ ┌─RMTMQMNAME(*SAME)────────────────────────────────┐
►─┴─TGTQNAME(target-queue-name)(3)─┘ └─RMTQNAME(─┬─*NONE─────────────┬─)(4)─┘ └─RMTMQMNAME(remote-queue-manager-name)(4)─┘─►
                                                 └─remote-queue-name─┘

  ┌─TMQNAME(*SAME)─────────────────────────────┐ ┌─DFNTYPE(*SAME)──────────────────┐ ┌─HIGHTHLD(*SAME)──────┐ ┌─LOWTHLD(*SAME)──────┐
►─┴─TMQNAME(transmission-queue-name)(4)─┘ └─DFNTYPE(─┬─*TEMPDYN─┬─)(5)─┘ └─HIGHTHLD(*0-100)(2)─┘ └─LOWTHLD(*0-100)(2)─┘─►
                                                     └─*PERMDYN─┘

  ┌─FULLEVT(*SAME)───────────┐ ┌─HIGHEVT(*SAME)───────────┐ ┌─LOWEVT(*SAME)───────────┐ ┌─SRVITV(*SAME)──────────────┐
►─┴─FULLEVT(─┬─*NO──┬─)(2)─┘ └─HIGHEVT(─┬─*NO──┬─)(2)─┘ └─LOWEVT(─┬─*NO──┬─)(2)─┘ └─SRVITV(0-999999999)(2)─┘────►
             └─*YES─┘                   └─*YES─┘                  └─*YES─┘

  ┌─SRVEVT(*SAME)─────────────┐ ┌─DISTLIST(*SAME)───────────┐
►─┴─SRVEVT(─┬─*HIGH─┬─)(2)─┘ └─DISTLIST(─┬─*NO──┬─)(2)─┘──────────────────────────────────────────────────────►◄
            ├─*OK───┤                    └─*YES─┘
            └─*NONE─┘
```

Notes:

1 Valid only for local, model, and alias queues.

2 Valid only for local and model queues.

3 Valid only for alias queues.

4 Valid only for remote queues.

5 Valid only for model queues.

P All parameters preceding this point can be specified positionally.

Purpose

The Copy MQM 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.

Required parameters

FROMQ

Specifies the name of the existing queue definition, to provide values for the attributes not specified in this command.

from-queue-name: Specify the name of the source MQM queue.

TOQ

Specifies the name of the new queue definition. The name can contain up to 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).

to-queue-name: Specify the name of the MQM queue being created.

Optional parameters

REPLACE

Specifies whether the new queue should replace an existing queue definition with the same name and type.

***NO**: Do not replace the existing queue definition. The command fails if the named queue already exists.

***YES**: Replace the existing queue definition provided that the conditions that require the use of the FORCE(*YES) option with the CHGMQMQ command do not apply. If a definition does not exist, one is created.

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

Specifies text that briefly describes the queue definition.

***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).

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.

***SAME**: The attribute is unchanged.

***NO**: Messages cannot be added to the queue.

***YES**: Messages can be added to the queue by authorized applications.

DFTPTY

Specifies the default priority of messages put on the queue.

***SAME**: The attribute is unchanged.

priority-value: Specify the default priority for the queue. The value must be in the range zero through the maximum priority supported, that is, (9).

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.

***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.

PRCNAME
Specifies the local name of the MQM process, for a local or model (*LCL or *MDL) queue, 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.

***SAME**: The attribute is unchanged.

***NONE**: The process name is blank.

process-name: Specify the name of the MQM process.

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.

***SAME**: The attribute is unchanged.

***NO**: Do not write trigger messages to the initiation queue.

***YES**: Triggering is active; trigger messages are written to the initiation queue.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

***SAME**: The attribute is unchanged.

***NO**: Applications cannot retrieve messages from the queue.

***YES**: Suitably authorized applications can retrieve messages from the queue.

| This parameter cannot be specified for a remote (*RMT) queue.

SHARE
Specifies whether multiple instances of applications, can open this queue for input,

***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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

DFTSHARE
Specifies the default share option for applications opening this queue for input.

***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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

MSGDLYSEQ
Specifies the message delivery sequence.

***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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

HDNBKTCNT
Specifies whether the count of backed out messages should be saved (hardened) across restarts of the message queue manager.

***SAME**: The attribute is unchanged.

***NO**: The backout count is not hardened.

***YES**: The backout count is hardened.

Note: On MQSeries for AS/400 the count is *always* hardened, regardless of the setting of this attribute.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

***SAME**: The attribute is unchanged.

***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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

TRGDEPTH
Specifies, for TRIGTYPE(*DEPTH), the number of messages that initiates a trigger message to the initiation queue.

Note: An application program can issue a call to MQSET to change the value of this attribute.

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

depth-value: Specify a value in the range 1 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

*SAME: The attribute is unchanged.

priority-value: Specify a value in the range of priority values that are supported, that is, 0 through 9.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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 that is started by the monitor.

For a transmission queue, you can use this parameter to specify the name of the channel to be started.

Note: An application program can issue a call to MQSET to change the value of this attribute.

*SAME: The attribute is unchanged.

***NONE**: No trigger data is specified.

trigger-data: Specify up to 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).

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

RTNITV
Specifies the retention interval. This interval is the number of hours for which the queue may 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 may be used to determine when a queue is no longer required.

Note: The message queue manager does not delete queues nor does it prevent queues from being deleted if their retention interval has not expired. It is the user's responsibility to take any required action.

*SAME: The attribute is unchanged.

interval-value: Specify a value in the range 0 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

MAXDEPTH

Specifies the maximum number of messages allowed on the queue. However, other factors may 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.

***SAME**: The attribute is unchanged.

depth-value: Specify a value in the range 0 through 640 000. However, if you are using the MQSeries for AS/400 Administration Utility with a platform other than AS/400, the limits are 0 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

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

***SAME**: The attribute is unchanged.

length-value: Specify a value in the range 0 through 4 194 304.

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

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

BKTTHLD

Specifies the backout threshold. Apart from allowing this attribute to be queried, MQSeries for AS/400 takes no action based on the value of the attribute.

***SAME**: The attribute is unchanged.

threshold-value: Specify a value in the range 0 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

BKTQNAME

Specifies the backout-queue name. Apart from allowing this attribute to be queried, MQSeries for AS/400 takes no action based on the value of the attribute.

***SAME**: The attribute is unchanged.

***NONE**: No backout queue is specified.

backout-queue-name: Specify the backout queue name.

This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

INITQNAME
Specifies the name of the initiation queue. That is, the local queue for trigger messages relating to this new, or changed, queue.

Note: The initiation queue must be on the same queue manager.

<u>*SAME</u>: The attribute is unchanged.

***NONE**: No initiation queue is specified.

initiation-queue-name: Specify the initiation queue name.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

USAGE
Specifies whether the queue is for normal usage, or for transmitting messages to a remote message queue manager.

<u>*SAME</u>: 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 the *MQSeries Intercommunication* book.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

TGTQNAME
Specifies the name of the local or remote target queue, for which this queue is an alias.

Note: The target queue does not need to exist at this time but it must exist when a process attempts to open the alias queue.

<u>*SAME</u>: The attribute is unchanged.

target-queue-name: Specify the name of the target queue.

RMTQNAME
Specifies the name of the queue on the remote system. If this definition is used for a queue-manager alias definition, RMTQNAME must be blank when the open occurs (*NONE can be specified).

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.

<u>*SAME</u>: 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.

| This parameter can be specified for a remote (*RMT) queue only

RMTMQMNAME
Specifies the name of the queue manager on the remote system.

If an application opens the local definition of a remote queue, RMTMQMNAME must not be the name of the local 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 local 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.

<u>*SAME</u>: The attribute is unchanged.

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

Note: The name is not checked to ensure that it contains only those characters normally allowed for queue manager names.

| This parameter can be specified for a remote (*RMT) queue only.

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 instead 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 local queue manager.

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

<u>*SAME</u>: 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.

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

The possible values are:

<u>*SAME</u>

The attribute is unchanged.

threshold-value

Specify a value in the range 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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 in the range 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

| This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

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.

This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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:

<u>*SAME</u>

The attribute is unchanged.

interval-value

Specify a value in the range 0 through 999 999 999. The value is in units of milliseconds.

This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

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 at least the time indicated by the SRVITV parameter.

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:

<u>*SAME</u>

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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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 applies to a model queue definition only.

The possible values are:

<u>*SAME</u>

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.

DISTLIST
Specifies whether or not the queue supports distribution lists

The possible values are:

<u>*SAME</u>

The attribute is unchanged.

***NO**

Distribution lists are not supported.

***YES**

Distribution lists are supported.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

CRTMQM (Create Message Queue Manager) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
                                           ┌─TEXT(*BLANK)──────┐ ┌─TRIGITV(999999999)──────┐
►►──CRTMQM──MQMNAME(queue-manager-name)─(P)┼───────────────────┼─┼─────────────────────────┼──────────────────►
                                           └─TEXT(description)─┘ └─TRIGITV(interval-value)─┘

   ┌─UDLMSGQ(*NONE)──────────────────────────────┐ ┌─DFTTMQ(*NONE)──────────────────────────────┐ ┌─MAXHDL(256)──────────────────────┐
►──┼─────────────────────────────────────────────┼─┼────────────────────────────────────────────┼─┼──────────────────────────────────┼──►
   └─UDLMSGQ(undelivered-message-queue-name)─────┘ └─DFTTMQ(default-transmission-queue-name)────┘ └─MAXHDL(maximum-handle-limit)─────┘

   ┌─MAXUMSG(10000)─────────────────────────┐
►──┼────────────────────────────────────────┼──────────────────────────────────────────────────────────────────►◄
   └─MAXUMSG(maximum-uncommitted-messages)──┘
```

Note:
P   All parameters preceding this point can be specified positionally.

Purpose

The Create Message Queue Manager (CRTMQM) command creates a local queue manager that can be started with the Start Message Queue Manager (STRMQM) command.

Required parameters

MQMNAME

Specifies the name of the message queue manager. 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).

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

Optional parameters

TEXT

Specifies text that briefly describes the queue-manager definition.

<u>*BLANK</u>: 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).

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.

<u>999999999</u>: The trigger time interval is 999 999 999 milliseconds.

interval-value: Specify a value in the range 0 through 999 999 999.

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.

*NONE: There is no undelivered-message queue. The attribute is set to a blank string.

undelivered-message-queue-name: Specify the name of the local queue that is to be used as the undelivered-message 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.

*NONE There is no default transmission queue. The attribute is set to a blank string.

default-transmission-queue-name: Specify the name of the local queue.

MAXHDL
Specifies the maximum number of handles that any one job can have open at the same time.

_256_: The default number of open handles is 256.

maximum-handle-limit: Specify a value in the range 0 through 999 999 999.

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.

_10000_: The default value is 10 000 uncommitted messages.

maximum-uncommitted-messages: Specify a value in the range 1 through 10 000.

Once you have created a queue manager with this command, use the CHGMQM command to change the default values. See “CHGMQM (Change Message Queue Manager) Command” on page 107 for further details.

CRTMQMCHL (Create MQM Channel) Command

```
                                                                          Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                                    ┌REPLACE(*NO)──┐  ┌TRPTYPE(*SYSDFTCHL)──────┐
►►──CRTMQMCHL──CHLNAME(channel-name)──CHLTYPE(─┬*SDR───┬)(P)──┴REPLACE(*YES)─┘──┴TRPTYPE(─┬*LU62─┬─)─┘(S)──────►
                                               ├*SVR───┤                                   └*TCP──┘
                                               ├*RCVR──┤
                                               ├*RQSTR─┤
                                               └*SVRCN─┘

  ┌TEXT(*SYSDFTCHL)──────────┐  ┌CONNAME(*SYSDFTCHL)────────────────┐  ┌TMQNAME(*SYSDFTCHL)─────────────────────┐
►─┴TEXT(─┬*BLANK──────┬─)────┘──┴CONNAME(─┬*NONE───────────┬─)(1)───┘──┴TMQNAME(transmission-queue-name)(2)─────┘──►
         └description─┘                    └connection-name─┘

  ┌MCANAME(*SYSDFTCHL)──────────────────────┐  ┌MCAUSERID(*SYSDFTCHL)──────────────┐  ┌BATCHSIZE(*SYSDFTCHL)──┐
►─┴MCANAME(─┬*NONE───────────────────────┬(3)─)┘──┴MCAUSERID(─┬*NONE──────────────┬─)┘──┴BATCHSIZE(batch-size)(6)┘─►
            └library-name/program-name───┘                   ├*PUBLIC────────────┤
                                                             └mca-user-identifier┘

  ┌DSCITV(*SYSDFTCHL)──────────────┐  ┌SHORTTMR(*SYSDFTCHL)──────────────┐  ┌SHORTRTY(*SYSDFTCHL)────────────┐
►─┴DSCITV(disconnect-interval)(4)──┘──┴SHORTTMR(short-retry-interval)(4)─┘──┴SHORTRTY(short-retry-count)(4)─┘──►

  ┌LONGTMR(*SYSDFTCHL)─────────────┐  ┌LONGRTY(*SYSDFTCHL)───────────┐  ┌SCYEXIT(*SYSDFTCHL)─────────────────────┐
►─┴LONGTMR(long-retry-interval)(4)─┘──┴LONGRTY(long-retry-count)(4)──┘──┴SCYEXIT(─┬*NONE──────────────────────┬─)┘──►
                                                                                  └library-name/program-name─┘

  ┌SCYUSRDATA(*SYSDFTCHL)───────────────────┐  ┌SNDEXIT(*SYSDFTCHL)──────────────────────────┐
►─┴SCYUSRDATA(─┬*NONE────────────────────┬─)┘──┴SNDEXIT(─┬*NONE──────────────────────────┬─)┘──────────────────────►
               └security-exit-user-data──┘               │ ┌─,──────────────────────────┐│
                                                         └─▼─library-name/program-name──┴┘

  ┌SNDUSRDATA(*SYSDFTCHL)───────────────────┐  ┌RCVEXIT(*SYSDFTCHL)──────────────────────────────────────┐
►─┴SNDUSRDATA(─┬*NONE────────────────────┬─)┘──┤                                                         ├──────────►
               │ ┌─,────────────────────┐│     │          ┌─,─────────────────────────────────┐          │
               └─▼─send-exit-user-data──┴┘     └RCVEXIT(──▼──*NONE──library-name/program-name──┴────)────┘

  ┌RCVUSRDATA(*SYSDFTCHL)──────────────────────┐  ┌MSGEXIT(*SYSDFTCHL)─────────────────────────────────┐
►─┴RCVUSRDATA(─┬*NONE───────────────────────┬─)┘──┴MSGEXIT(─┬*NONE────────────────────────────────┬─)─┘──────────►
               │ ┌─,───────────────────────┐│                │ ┌─,──────────────────────────┐       │
               └─▼─receive-exit-user-data──┴┘                └─▼─library-name/program-name──┴(6)───┘

  ┌MSGUSRDATA(*SYSDFTCHL)───────────────────────────┐  ┌MSGRTYEXIT(*SYSDFTCHL)───────────────────────────┐
►─┴MSGUSRDATA(─┬*NONE────────────────────────────┬─)┘──┴MSGRTYEXIT(─┬*NONE──────────────────────┬─)(5)───┘──────►
               │ ┌─,───────────────────────┐     │                  └library-name/program-name─┘
               └─▼─message-exit-user-data──┴(6)──┘

  ┌MSGRTYDATA(*SYSDFTCHL)──────────────────────────┐  ┌MSGRTYNBR(*SYSDFTCHL)─────────────────┐  ┌CVTMSG(*SYSDFTCHL)───────┐
►─┴MSGRTYDATA(─┬*NONE─────────────────────────┬─)(5)┘──┴MSGRTYNBR(message-retry-number)(5)──┘──┴CVTMSG(─┬*YES─┬─)(4)─┘──►
               └message-retry-exit-user-data──┘                                                         └*NO──┘

  ┌MSGRTYITV(*SYSDFTCHL)──────────────────────┐  ┌PUTAUT(*SYSDFTCHL)──────┐  ┌SEQNUMWRAP(*SYSDFTCHL)─────────────────────────┐
►─┴MSGRTYITV(message-retry-interval)(5)───────┘──┴PUTAUT(─┬*DFT─┬─)(5)────┘──┴SEQNUMWRAP(sequence-number-wrap-value)(6)─┘──►
                                                          └*CTX─┘

  ┌MAXMSGLEN(*SYSDFTCHL)───────────────────┐  ┌HRTBTINTVL(*SYSDFTCHL)──────────────┐  ┌NPMSPEED(*SYSDFTCHL)──────────┐
►─┴MAXMSGLEN(maximum-message-length)───────┘──┴HRTBTINTVL(heart-beat-interval)─────┘──┴NPMSPEED(─┬*FAST───┬─)(6)──┘──►◄
                                                                                                  └*NORMAL─┘
```

Notes:

1 Optional only for CHLTYPE(*RCVR or *SVR), otherwise required.

2 Required for CHLTYPE(*SDR or *SVR), otherwise not valid.

3 Valid only for CHLTYPE(*SDR or *SVR or *RQSTR).

4 Valid only for CHLTYPE(*SDR or *SVR).

5 Valid only for CHLTYPE(*RCVR or *RQSTR).

6 Valid only for CHLTYPE(*SDR, *SVR, *RCVR, or *RQSTR).

P All parameters preceding this point can be specified positionally.

S The value *SYSDFTCHL is that defined in the system default channel for the type of channel being defined (that is, one of SYSTEM.DEF.SENDER, SYSTEM.DEF.SERVER, SYSTEM.DEF.RECEIVER, SYSTEM.DEF.REQUESTER, and SYSTEM.DEF.SVRCONN). It can be changed by the installation.

Purpose

The Create MQM Channel (CRTMQMCHL) command creates a new MQM channel definition, specifying those attributes that are to be different from the default values.

Restriction

To use this command, you must be signed on as QPGMR, or QSYSOPR, or have *ALLOBJ authority.

Required parameters

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.

CHLTYPE

Specifies the type of the channel being defined.

***SDR**: Sender channel

***SVR**: Server channel

***RCVR**: Receiver channel

***RQSTR**: Requester channel

***SVRCN**: Server connection

TRPTYPE

Specifies the transmission protocol.

***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).

Optional parameters

REPLACE

Specifies whether the new channel definition should replace an existing channel definition that has the same name.

***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.

TEXT

Specifies text that briefly describes the channel definition.

***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).

CONNAME
Specifies the name of the machine to be connected.

***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 name of the machine.

*LU62. This is the name of the CSI object.

*TCP. This is either the name of the host or the network address of the remote machine.

This parameter is valid for channel types (CHLTYP) *SDR, *SVR, and *RQSTR. It is required for channel types of *SDR or *RQSTR, and optional for a channel type of *SVR.

| This parameter cannot be specified for channel type (CHLTYPE) of *SVRCN.

TMQNAME
Specifies the name of the transmission queue.

***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.

This parameter is required if the channel type (CHLTYPE) is *SDR or *SVR. For other channel types, the parameter must not be specified.

MCANAME
This parameter is reserved and should not be used.

***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 or *SVRCN.

MCAUSERID
Message channel agent user identifier. 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 of this attribute is taken from the system default channel for the type of 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.

BATCHSIZE
Specifies the maximum number of messages that should be sent down a channel before a checkpoint is taken.

***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 a channel type (CHLTYPE) of *SVRCN.

DSCITV
Specifies the disconnect interval for a sender or server (*SDR or *SVR) channel. This defines the maximum number of seconds that the channel waits for messages to be put on a transmission queue before ending the channel.

***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 999 999. Zero means an indefinite wait.

This parameter cannot be specified for channel type (CHLTYPE) of *RCVR.

SHORTTMR
Specifies the short retry wait interval for a sender or server (*SDR or *SVR) channel that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine.

***SYSDFTCHL**: The value of this attribute is taken from the system default channel for the type of channel being created.

short-retry-interval: Specify a value in the range 0 through 999 999 999.

This parameter cannot be specified for channel type (CHLTYPE) of *RCVR.

SHORTRTY
Specifies the short retry count for a sender or server (*SDR or *SVR) channel 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.

***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 999 999 999. A value of zero means that no retries are allowed.

This parameter cannot be specified for channel type (CHLTYPE) of *RCVR.

LONGTMR
Specifies the long retry wait interval for a sender or server (*SDR or *SVR) channel that is started automatically by the channel initiator. It defines the interval in seconds between attempts to establish a connection to the remote machine, after the count specified by SHORTRTY has been exhausted.

***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 999 999 999.

This parameter cannot be specified for channel type (CHLTYPE) of *RCVR.

LONGRTY

Specifies the long retry count for a sender or server (*SDR or *SVR) channel 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.

<u>*SYSDFTCHL</u>: 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 999 999 999. A value of zero means that no retries are allowed.

This parameter cannot be specified for channel type (CHLTYPE) of *RCVR.

SCYEXIT

Specifies the entry point 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 begin 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.

<u>*SYSDFTCHL</u>: The value of this attribute is taken from the system default channel of the specified type.

***NONE**: The security exit program is not invoked.

library-name/program-name: Specify the fully-qualified name for the security exit program.

SCYUSRDATA

Specifies a maximum of 32 characters of user data that is passed to the channel security exit program.

<u>*SYSDFTCHL</u>: 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.

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, and the contents of the buffer can be modified as required.

| You can specify the names of up to 10 exit programs by specifying multiple
| strings separated by commas.

<u>*SYSDFTCHL</u>: The value of this attribute is taken from the system default channel of the specified type.

***NONE**: The send exit is not invoked.

library-name/program-name: Specify the fully-qualified name for the send exit program.

SNDUSRDATA

Specifies a maximum of 32 characters of user data that is passed to the send exit program.

You can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first message exit specified, the second string to the second exit, and so on.

***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 that is passed to the send exit program.

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.

You can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

***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.

library-name/program-name: Specify the fully-qualified name for the receive exit program.

RCVUSRDATA

Specifies user data that is passed to the receive exit.

***SYSDFTCHL**: The value of this attribute is taken from the system default channel of the specified type.

You can specify up to 10 strings, each of length 32 characters. The first string of data is passed to the first message exit specified, the second string to the second exit, and so on.

***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.

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.

You can specify the names of up to 10 exit programs by specifying multiple strings separated by commas.

***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.

library-name/program-name: Specify the fully-qualified name for the message exit program.

| This parameter cannot be specified for a server-connection (*SVRCN) channel.

MSGUSRDATA
Specifies user data that is passed to the message exit.

| You can specify up to 10 strings, each of length 32 characters. The first string
| of data is passed to the first message exit specified, the second string to the
| second exit, and so on.

***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 a server-connection (*SVRCN) channel.

MSGRTYDATA
Specifies that user data is passed to the message retry exit program.

***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 that is passed to the message retry exit program.

MSGRTYEXIT
Specifies the entry point of the program to be called as the message retry exit for a receiver or requester (*RCVR, *RQSTR) channel.

***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.

library-name/program-name: Specify the fully qualified name for the message retry exit program.

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.

This is valid for a receiver or requester (*RCVR, *RQSTR) channel only.

***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 999 999 999. A value of zero signifies that the retry will be performed as soon as possible.

MSGRTYNBR

Specifies the number of times the channel retries before it concludes 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.

***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 999 999 999. A value of zero signifies that no retries will be performed.

PUTAUT

Specifies whether the user identifier in the context information associated with a message should be used to establish authority to put the message on the destination queue. This applies only to receiver and requester (*RCVR and *RQSTR) channels.

***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 type (CHLTYPE) of *RCVR.

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.

***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 999 999 999.

| This parameter cannot be specified for a server-connection (*SVRCN) channel.

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.

***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 4 194 304. A value of zero signifies that the maximum length is unlimited.

CVTMSG

Specifies whether the application data in the message should be converted before the message is transmitted. This applies only to sender and server (*SDR and *SVR) channels.

<u>*SYSDFTCHL</u> 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 type (CHLTYPE) of *RCVR.

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:

<u>*SYSDFTCHL</u> The value of the attribute is taken from the system default channel of the specified type.

heart-beat-interval: Specify a value in the range 0 through 999 999 999. A value of zero signifies that no retries will be performed.

Note: For implementation reasons, the maximum heartbeat interval that can be used is 999 999. Values exceeding this are treated as 999 999. A value of zero means that no heartbeat exchanges are to take place.

NPMSPEED

Specifies whether the channel supports fast-nonpersistent messages. This is only valid for sender, server, receiver, or requester (*SDR, *SVR, *RCVR, *RQSTR) channels.

The possible values are:

<u>*SYSDFTCHL</u> The value of the attribute is taken from the system default channel of the specified type.

***FAST** The channel supports fast nonpersistent messages.

***NORMAL** The channel does not support fast nonpersistent messages.

CRTMQMPRC (Create MQM Process) Command

```
                                                             Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                               ┌─REPLACE(*NO)──┐ ┌─TEXT(*SYSDFTPRC)──────┐
►►──CRTMQMPRC──PRCNAME(process-name)─(P)───────┼───────────────┼─┼───────────────────────┼──────────────────►
                                               └─REPLACE(*YES)─┘ └─TEXT(─┬─*BLANK──────┬─)─┘
                                                                         └─description─┘

   ┌─APPTYPE(*SYSDFTPRC)─(1)─────┐ ┌─APPID(*SYSDFTPRC)─(1)───┐ ┌─USRDATA(*SYSDFTPRC)─(1)─┐
►──┼─────────────────────────────┼─┼─────────────────────────┼─┼─────────────────────────┼────────────────►
   └─APPTYPE(─┬─*OS400─────┬─)───┘ └─APPID(application-id)───┘ └─USRDATA(─┬─*NONE─────┬─)┘
              ├─*CICS──────┤                                              └─user-data─┘
              └─user-value─┘

   ┌─ENVDATA(*SYSDFTPRC)─(1)──────────────┐
►──┼──────────────────────────────────────┼──────────────────────────────────────────────────────────────►◄
   └─ENVDATA(─┬─*NONE────────────────┬─)──┘
              └─environment-data─────┘
```

Notes:

1 The value used is that defined in the system default process object, that is, SYSTEM.DEFAULT.PROCESS.

P All parameters preceding this point can be specified positionally.

Purpose

The Create MQM Process (CRTMQMPRC) command creates a new MQM process definition, specifying those attributes that are to be different from the default.

Required parameters

PRCNAME

The name of the new MQM process definition to be created.

process-name: Specify the name of the new MQM process definition. The name can contain up to 48 characters.

Optional parameters

REPLACE

If a process definition with the same name already exists, this specifies whether it is to be replaced.

***NO**: This definition does not replace any existing process definition with the same name. The command fails if one already exists.

***YES**: This definition replaces any existing process definition with the same name.

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).

***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.

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

<u>*SYSDFTPRC</u>: The value of this attribute is taken from the system default process.

***OS400**: Represents an OS/400 application.

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

user-value: User-defined application type in the recommended range 65 536-999 999 999.

Note: The values within this range are not tested, and any other value will be accepted.

APPID
Application identifier. This is the name of the application to be started, on the platform for which the command is processing, and is typically be a program name and library name.

<u>*SYSDFTPRC</u>: The value of this attribute is taken from the system default process.

application-id: The maximum length is 256 characters.

USRDATA
A character string that contains user information pertaining to the application, as defined by APPID, to be started.

<u>*SYSDFTPRC</u>: The value of 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.

ENVDATA
A character string that contains environment information pertaining to the application, as defined by APPID, to be started.

<u>*SYSDFTPRC</u>: The value of this attribute is taken from the system default process.

***NONE**: The environment data is blank.

environment-data: The maximum length is 128 characters.

CRTMQMQ (Create MQM Queue) Command

```
                                                                              Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                                 ┌─REPLACE(*NO)──┐  ┌─TEXT(*SYSDFTQ)──────────────┐      ┌─PUTENBL(*SYSDFTQ)──┐
►►──CRTMQMQ──QNAME(queue-name)──QTYPE(──┬─*ALS─┬─)──(P)──┼────────────────┼──┼──────────────────────────────┼─(S)──┼─────────────────────┼──►
                                        ├─*LCL─┤          └─REPLACE(*YES)─┘  └─TEXT(──┬─*BLANK──────┬─)─┘      └─PUTENBL(──┬─*NO──┬─)─┘
                                        ├─*RMT─┤                                     └─description─┘                    └─*YES─┘
                                        └─*MDL─┘

   ┌─DEFPTY(*SYSDFTQ)──────┐  ┌─DFTMSGPST(*SYSDFTQ)──┐  ┌─PRCNAME(*SYSDFTQ)──────────────┐  ┌─TRGENBL(*SYSDFTQ)────┐
►──┼────────────────────────┼──┼──────────────────────┼──┼─────────────────────────────────┼──┼──────────────────────┼──►
   └─DEFPTY(priority-value)─┘  └─DFTMSGPST(──┬─*NO──┬─)─┘  └─PRCNAME(──┬─*NONE────────┬─)──(2)─┘  └─TRGENBL(──┬─*NO──┬─)──(2)─┘
                                             └─*YES─┘                 └─process-name─┘                      └─*YES─┘

   ┌─GETENBL(*SYSDFTQ)──────┐  ┌─SHARE(*SYSDFTQ)──────┐  ┌─DFTSHARE(*SYSDFTQ)───┐  ┌─MSGDLYSEQ(*SYSDFTQ)───────┐
►──┼─────────────────────────┼──┼───────────────────────┼──┼───────────────────────┼──┼────────────────────────────┼──►
   └─GETENBL(──┬─*NO──┬─)──(1)─┘  └─SHARE(──┬─*NO──┬─)──(2)─┘  └─DFTSHARE(──┬─*NO──┬─)──(2)─┘  └─MSGDLYSEQ(──┬─*PTY──┬─)──(2)─┘
               └─*YES─┘                     └─*YES─┘                        └─*YES─┘                    └─*FIFO─┘

   ┌─HDNBKTCNT(*SYSDFTQ)─────┐  ┌─TRGTYPE(*SYSDFTQ)──────────┐  ┌─TRGDEPTH(*SYSDFTQ)─────────┐  ┌─TRGMSGPTY(*SYSDFTQ)──────────────┐
►──┼──────────────────────────┼──┼─────────────────────────────┼──┼─────────────────────────────┼──┼───────────────────────────────────┼──►
   └─HDNBKTCNT(──┬─*NO──┬─)──(2)─┘  └─TRGTYPE(──┬─*FIRST─┬─)──(2)─┘  └─TRGDEPTH(depth-value)──(2)─┘  └─TRGMSGPTY(priority-value)──(2)─┘
                 └─*YES─┘                      ├─*ALL───┤
                                               ├─*DEPTH─┤
                                               └─*NONE──┘

   ┌─TRGDATA(*SYSDFTQ)────────────────┐  ┌─RTNINTV(*SYSDFTQ)────────────┐  ┌─MAXDEPTH(*SYSDFTQ)──────────┐
►──┼───────────────────────────────────┼──┼───────────────────────────────┼──┼──────────────────────────────┼────────────────────────────►
   └─TRGDATA(──┬─*NONE────────┬─)──(2)─┘  └─RTNINTV(interval-value)──(2)─┘  └─MAXDEPTH(depth-value)──(2)─┘
               └─trigger-data─┘

   ┌─MAXMSGLEN(*SYSDFTQ)────────────┐  ┌─BKTTHLD(*SYSDFTQ)─────────────┐  ┌─BKTQNAME(*SYSDFTQ)───────────────────────┐
►──┼─────────────────────────────────┼──┼────────────────────────────────┼──┼───────────────────────────────────────────┼──────────────►
   └─MAXMSGLEN(length-value)──(2)─┘  └─BKTTHLD(threshold-value)──(2)─┘  └─BKTQNAME(──┬─*NONE──────────────┬─)──(2)─┘
                                                                                     └─backout-queue-name─┘

   ┌─INITQNAME(*SYSDFTQ)──────────────────────────┐  ┌─USAGE(*SYSDFTQ)──────────┐  ┌─TGTQNAME(*SYSDFTQ)───────────────────┐
►──┼───────────────────────────────────────────────┼──┼───────────────────────────┼──┼───────────────────────────────────────┼───────►
   └─INITQNAME(──┬─*NONE─────────────────┬─)──(2)─┘  └─USAGE(──┬─*NORMAL─┬─)──(2)─┘  └─TGTQNAME(target-queue-name)──(3)─┘
                 └─initiation-queue-name─┘                      └─*TMQ────┘

   ┌─RMTQNAME(*SYSDFTQ)────────────────────────┐  ┌─RMTMQMNAME(*SYSDFTQ)─────────────────────────┐
►──┼────────────────────────────────────────────┼──┼───────────────────────────────────────────────┼──────────────────────────────────►
   └─RMTQNAME(──┬─*NONE─────────────┬─)──(4)─┘  └─RMTMQMNAME(remote-queue-manager-name)──(4)─┘
                └─remote-queue-name─┘

   ┌─TMQNAME(*SYSDFTQ)──────────────────────────┐  ┌─DFNTYPE(*SYSDFTQ)─────────────┐  ┌─HIGHTHLD(*SYSDFTQ)──┐  ┌─LOWTHLD(*SYSDFTQ)──┐
►──┼─────────────────────────────────────────────┼──┼────────────────────────────────┼──┼──────────────────────┼──┼─────────────────────┼──►
   └─TMQNAME(transmission-queue-name)──(4)─┘  └─DFNTYPE(──┬─*TEMPDYN─┬─)──(5)─┘  └─HIGHTHLD(*0-100)──(2)─┘  └─LOWTHLD(*0-100)──(2)─┘
                                                          └─*PERMDYN─┘

   ┌─FULLEVT(*SYSDFTQ)──────┐  ┌─HIGHEVT(*SYSDFTQ)──────┐  ┌─LOWEVT(*SYSDFTQ)──────┐  ┌─SRVITV(*SYSDFTQ)─────────────┐
►──┼─────────────────────────┼──┼─────────────────────────┼──┼────────────────────────┼──┼───────────────────────────────┼─────────►
   └─FULLEVT(──┬─*NO──┬─)──(2)─┘  └─HIGHEVT(──┬─*NO──┬─)──(2)─┘  └─LOWEVT(──┬─*NO──┬─)──(2)─┘  └─SRVITV(0-999999999)──(2)─┘
               └─*YES─┘                      └─*YES─┘                     └─*YES─┘

   ┌─SRVEVT(*SYSDFTQ)────────┐  ┌─DISTLIST(*SYSDFTQ)─────┐
►──┼──────────────────────────┼──┼─────────────────────────┼──────────────────────────────────────────────────────────────────────►◄
   └─SRVEVT(──┬─*HIGH─┬─)──(2)─┘  └─DISTLIST(──┬─*NO──┬─)──(2)─┘
              ├─*OK───┤                        └─*YES─┘
              └─*NONE─┘
```

Notes:

1 Valid only for local, model, and alias queues.
2 Valid only for local and model queues.
3 Valid only for alias queues.
4 Valid only for remote queues.
5 Valid only for model queues.
P All parameters preceding this point can be specified positionally.
S The value *SYSDFTQ is that defined in the system default queue object for the queue being defined (that is, one of SYSTEM.DEFAULT.ALIAS.QUEUE, SYSTEM.DEFAULT.LOCAL.QUEUE, and SYSTEM.DEFAULT.REMOTE.QUEUE).

Purpose

The Create MQM 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.

Required parameters

QNAME

Specifies the name of the queue definition. Queue names must be unique. If a queue definition with this name already exists, REPLACE(*YES) must be specified.

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).

queue-name: Specify the name of the new MQM queue.

QTYPE

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.

***ALS**: An alias queue.

***LCL**: A local queue.

***MDL**: A model queue.

***RMT**: A remote queue.

Optional parameters

REPLACE

Specifies whether the new queue should replace an existing queue definition with the same name and type.

<u>*NO</u>: Do not replace the existing queue. The command fails if the named queue already exists.

***YES**: Replace the existing queue definition provided that the conditions which require the use of the FORCE(*YES) option with the CHGMQMQ command do not apply. If a definition does not exist, one is created.

Note: If the queue is a local queue, and a queue with the same name does already exist, any messages that are already on it are retained.

TEXT

Specifies text that briefly describes the queue definition.

<u>*SYSDFTQ</u>: The text 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).

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.

<u>*SYSDFTQ</u>: 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.

DFTPTY
Specifies the default priority of messages put on the queue.

***SYSDFTQ**: The default priority is taken from the system default queue of the specified type.

priority-value: Specify the default priority for the queue. The value must be in the range zero through the maximum priority supported, that is, (9).

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.

***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.

PRCNAME
Specifies the local name of the MQM process for a local or model (*LCL or *MDL) 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.

***SYSDFTQ**: The process name is taken from the system default queue of the specified type.

***NONE**: The process name is blank.

process-name: Specify the name of the MQM process.

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.

***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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

***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.

| This parameter cannot be specified with a remote (*RMT) queue.

SHARE
Specifies whether multiple instances of applications can open this queue for input.

***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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

DFTSHARE
Specifies the default share option for applications opening this queue for input.

***SYSDFTQ**: The value of this attribute is taken from the system default queue of the specified type.

***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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

MSGDLYSEQ
Specifies the message delivery sequence.

***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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

HDNBKTCNT
Specifies whether the count of backed out messages should be saved (hardened) across restarts of the message queue manager.

***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.

Note: On MQSeries for AS/400 the count is *always* hardened, regardless of the setting of this attribute.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

***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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

***SYSDFTQ**: The value of this attribute is taken from the system default queue of the specified type.

depth-value: Specify a value in the range 1 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

***SYSDFTQ**: The value of this attribute is taken from the system default queue for the relevant queue type.

priority-value: Specify a value in the range of supported priority values, that is, 0 through 9.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

For a transmission queue, you can use this parameter to specify the name of the channel to be started.

Note: An application program can issue a call to MQSET to change the value of this attribute.

<u>*SYSDFTQ</u>: 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.

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).

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

RTNITV
Specifies the retention interval. This interval is the number of hours for which the queue may 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 may be used to determine when a queue is no longer required.

Note: The message queue manager does not delete queues nor does it prevent queues from being deleted if their retention interval has not expired. It is the user's responsibility to take any required action.

<u>*SYSDFTQ</u>: The value of this attribute is taken from the system default queue of the specified type.

interval-value: Specify a value in the range 0 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

MAXDEPTH
Specifies the maximum number of messages allowed on the queue. However, other factors may 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.

<u>*SYSDFTQ</u>: The value of this attribute is taken from the system default queue of the specified type.

depth-value: Specify a value in the range 0 through 640 000. However, if you are using the MQSeries for AS/400 Administration Utility with a platform other than AS/400, the limits are 0 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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.

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

<u>*SYSDFTQ</u>: The value of this attribute is taken from the system default queue of the specified type.

length-value: Specify a value in the range 0 through 4 194 304.

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

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

BKTTHLD
Specifies the backout threshold. Apart from allowing this attribute to be queried, MQSeries for AS/400 takes no action based on the value of the attribute.

<u>*SYSDFTQ</u>: The value of this attribute is taken from the system default queue of the specified type.

threshold-value: Specify a value in the range 0 through 999 999 999.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

BKTQNAME
Specifies the backout queue name. Apart from allowing this attribute to be queried, MQSeries for AS/400 takes no action based on the value of the attribute.

<u>*SYSDFTQ</u>: The backout queue name is taken from the system default queue of the specified type.

***NONE**: No backout queue is specified.

backout-queue-name: Specify the backout queue name.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

INITQNAME
Specifies the name of the initiation queue. That is, the local queue for trigger messages relating to this new, or changed, queue.

Note: The initiation queue must be on the same queue manager.

<u>*SYSDFTQ</u>: The initiation queue name is taken from the system default queue of the specified type.

***NONE**: No initiation queue is specified.

initiation-queue-name: Specify the initiation queue name.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

USAGE
Specifies whether the queue is for normal usage or for transmitting messages to a remote message queue manager.

<u>*SYSDFTQ</u>: The value of this attribute is taken from the system default queue of the specified 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 *MQSeries Intercommunication* book.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

TGTQNAME

Specifies the name of the local or remote target queue, for which this queue is an alias.

You should 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.

Note: The target queue does not need to exist at this time but it must exist when a process attempts to open the alias queue.

<u>***SYSDFTQ**</u>: The name of the target queue is taken from the SYSTEM.DEFAULT.ALIAS.QUEUE.

target-queue-name: Specify the name of the target queue.

RMTQNAME

Specifies the name of the queue on the remote system.

If this definition is used for a queue-manager alias definition, RMTQNAME must be blank when the open occurs (*NONE can be specified).

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.

<u>***SYSDFTQ**</u>: 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 should be used (if it is not the default) 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.

| This parameter can be specified for a remote (*RMT) queue only.

RMTMQMNAME

Specifies the name of the queue manager on the remote system.

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.

<u>*SYSDFTQ</u>: 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: The name is not checked to ensure that it contains only those characters normally allowed for queue manager names.

| This parameter can be specified for a remote (*RMT) queue only.

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 local queue manager.

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

<u>*SYSDFTQ</u>: 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.

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

The possible values are:

<u>*SYSDFTQ</u>

The value of this attribute is taken from the system default queue of the specified type.

threshold-value

Specify a value in the range 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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

The possible values are:

<u>*SYSDFTQ</u>

The value of this attribute is taken from the system default queue of the specified type.

threshold-value

Specify a value in the range 0 through 100. This value is used as a percentage of the maximum queue depth (MAXDEPTH parameter).

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

FULLEVT
Specifies whether Queue Full events are generated.

The possible values are:

<u>*SYSDFTQ</u>

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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

HIGHEVT
Specifies whether Queue Depth High events are generated.

The possible values are:

<u>*SYSDFTQ</u>

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.

This parameter cannot be specified for a remote or alias (*RMT or *ALS) queue.

LOWEVT
Specifies whether Queue Depth Low events are generated.

The possible values are:

<u>*SYSDFTQ</u>

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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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 in the range 0 through 999 999 999. The value is in units of milliseconds.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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 at least the time indicated by the SRVITV parameter.

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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

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 applies to a model queue definition only.

The possible values are:

***SYSDFTQ**

The value of this attribute is taken from the system default queue of the specified type.

***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.

DISTLIST

Specifies whether or not 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.

| This parameter cannot be specified for a remote or alias (*RMT or *ALS)
| queue.

CVTMQMDTA (Convert MQM Data Type) Command

```
                                                          Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

►►──CVTMQMDTA──FROMFILE(─┬─*LIBL/──────────────┬──────from-file-name───)──FROMMBR(from-member-name)─(P)──────────────►
                         ├─*CURLIB/────────────┤
                         └─from-library-name/──┘

►─┬─TOFILE(*LIBL/)───────────────────────────────────────────┬─┬─TOMBR(*FROMMBR)──────────┬─┬─RPLTOMBR(*YES)─┬──────────►◄
  └─TOFILE(─┬─*CURLIB/────────────┬─┬─QCSRC──────────┬─)─────┘ └─TOMBR(to-member-name/)───┘ └─RPLTOMBR(*NO)──┘
            └─to-library-name/────┘ └─to-file-name───┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Convert MQM Data Type (CVTMQMDTA) command produces a fragment of code that performs data conversion on data type structures, for use by the data-conversion exit program.

For information about the data-conversion exit program, see the *MQSeries Application Programming Guide*.

Support is provided for the C programming language only.

Required parameters

FROMFILE
Specifies the qualified name of the file, in the form LIBRARY/FILE, that contains the data to be converted.

<u>*LIBL</u>: 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 be converted.

FROMMBR
Specifies the name of the member containing the data to be converted.

from-member-name: Specify the name of the member.

Optional parameters

TOFILE
Specifies the qualified name of the file, in the form LIBRARY/FILE, that contains the converted data.

<u>*LIBL</u>: 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.

<u>QCSRC</u>: QCSRC is used.

to-file-name: Specify the name of the file.

TOMBR

Specifies the name of the member containing the converted data.

***FROMMBR**: The from member name is used.

to-member-name: Specify the name of the member containing the converted data.

RPLTOMBR

Specifies whether the converted data replaces the existing member.

***YES**: The converted data replaces the existing member.

***NO**: The converted data does not replace the existing member.

DLTMQM (Delete Message Queue Manager) Command

```
                                         Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──DLTMQM──MQMNAME(queue-manager-name)─(P)────────────────────────────────────────►◄
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Delete Message Queue Manager (DLTMQM) command deletes the local queue manager.

Required parameters

MQMNAME
Specifies the name of the message queue manager.

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

DLTMQMCHL (Delete MQM Channel) Command

```
                                                      Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──DLTMQMCHL──CHLNAME(channel-name)─(P)──────────────────────────────────────────────────────────►◄
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

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

Restriction

This command is shipped with public *EXCLUDE authority and with the QPGMR and QSYSOPR user profiles having private authority to use the command.

Required parameters

CHLNAME
Specifies the name of the channel definition.

DLTMQMPRC (Delete MQM Process) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
►►──DLTMQMPRC──PRCNAME(process-name)(P)──────────────────────────────────────►◄
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

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

Required parameters

PRCNAME
The name of the process definition to be deleted.

process-name: Specify the name of the process definition.

DLTMQMQ (Delete MQM Queue) Command

```
                                                      Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──DLTMQMQ──QNAME(queue-name)─(P)──────────────────────────────────────────────────────────────►◄
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Delete MQM Queue (DLTMQMQ) command deletes an MQM 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.

This command fails if an application has:

- This queue open
- A queue that resolves to this queue open
- A queue open that resolved through this definition as a queue-manager alias

An application using the definition as a reply-to queue alias, however, does not cause this command to fail.

Required parameters

QNAME
Specifies the name of the queue.

queue-name: Specify the name of the queue.

DSCMQM (Disconnect Message Queue Manager) Command

```
                                              Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──DSCMQM──────────────────────────────────────────────────────────────────────────────►◄
```

Purpose

The Disconnect Message Queue Manager (DSCMQM) command is used to disconnect from the Message Queue Manager. It should always be used immediately after a sequence of MQM commands, that is preceded by the CCTMQM command.

Required parameters

This command has no parameters.

DSPMQM (Display Message Queue Manager) Command

```
                                          Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
              ┌─OUTPUT(*)───────┐
►►──DSPMQM────┤                 ├(P)──────────────────────────────────────────────►◄
              └─OUTPUT(*PRINT)──┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Display Message Queue Manager (DSPMQM) command displays the attributes of the local queue manager. These may be any of those shown under Optional parameters.

Optional parameters

OUTPUT

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job’s spooled output.

<u>*</u>: 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.

DSPMQMAUT (Display MQM Authority) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
                                                     ┌─OUTPUT(*)───────┐
►►──DSPMQMAUT──OBJ(object-name(1))──OBJTYPE(──┬─*Q────┬──)──┤                 ├(P)──────────────►◄
                                          ├─*PRC──┤    └─OUTPUT(*PRINT)──┘
                                          ├─*MQM──┤
                                          ├─*ADM──┤
                                          └─*CTLG─┘
```

Notes:

1 Object name is a 48-character MQM queue-manager, queue or process name.

P All parameters preceding this point can be specified positionally.

Purpose

The Display MQM Authority (DSPMQMAUT) command shows, for the specified object, the list of authorized users of the object, and their authorities for the object. If the object is secured by an authorization list, the name of the authorization list is also shown.

The following are shown for the specified object:

- The 48-character MQM object name
- The MQM object type
- The AS/400 name, type, library, and owner of the MQM object
- The name of the authorization list securing the object
- A list of all the users who are authorized to use the object
- The authorities that each user has for the object

If the user entering the command does not have object management authority for the object, only that user's name and authorities are shown. The names of the other users, and their authorities for the object, are not shown.

Required parameters

OBJ

Specifies the name of the MQM object for which the authorized users and their authorities are displayed.

OBJTYPE

Specifies the object type.

<u>*Q</u>: All queue object types.

***PRC**: Process definition.

***MQM**: Message queue manager. This object holds the attributes of the message queue manager.

***ADM**: Message queue manager administration object. The administration object has the same name as the message queue manager object. It holds the MQM authorities to specify message context, and to open MQM objects as an alternate user.

***CTLG**: Message queue manager catalog object. The catalog object has the same name as the message queue manager object. It holds the names of MQM objects. A user needs authorities on this object to be able to start or stop the message queue manager, or to create or delete MQM queues and process definitions.

Optional parameters

OUTPUT

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

*****: 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.

DSPMQMCHL (Display MQM Channel) Command

```
                                                    Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                        ┌─OUTPUT(*)──────┐
►►──DSPMQMCHL──CHLNAME(channel-name)──┤                ├(P)──────────────────────────────────►◄
                                        └─OUTPUT(*PRINT)─┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Display MQM Channel (DSPMQMCHL) command displays the attributes of an existing MQM channel definition.

Required parameters

CHLNAME

Specifies the name of the channel definition.

channel-name: Specify the name of the channel definition.

Optional parameters

OUTPUT

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

*****: 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.

DSPMQMCSVR (Display MQM Command Server) Command

```
                                                   Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──DSPMQMCSVR──────────────────────────────────────────────────────────────────────────────►◄
```

Purpose

The Display MQM Command Server (DSPMQMCSVR) command displays the status of the MQM command server.

The status of the command server can be one of the following:

- Stopped
- Starting
- Running
- Running with the SYSTEM.ADMIN.COMMAND.QUEUE not enabled to get messages from the queue, that is, RUNNING GETENBL(*NO)
- Ending

Required parameters

This command has no parameters.

DSPMQMOBJN (Display MQM Object Names) Command

```
                                                          Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
                                                                      ┌─OUTPUT(*)───────┐
►►──DSPMQMOBJN──OBJ(──┬─*ALL───────────────────┬─)──OBJTYPE(──┬─*ALLMQM─┬─)──┤                 ├─(P)──────────────►◄
                      ├─generic*-object-name─(1)┤              ├─*ALL────┤    └─OUTPUT(*PRINT)──┘
                      └─object-name─(1)─────────┘              ├─*Q──────┤
                                                               ├─*ALSQ───┤
                                                               ├─*LCLQ───┤
                                                               ├─*RMTQ───┤
                                                               ├─*MDLQ───┤
                                                               ├─*PRC────┤
                                                               ├─*MQM────┤
                                                               ├─*ADM────┤
                                                               ├─*CTLG───┤
                                                               ├─*USRSPC─┤
                                                               └─*USRIDX─┘
```

Notes:

1 If OBJTYPE is specified as *ALL, *USRSPC or *USRIDX then object name and generic object name are 10-character OS/400 object or generic object names, otherwise they are 48-character MQM object or generic object names.

P All parameters preceding this point can be specified positionally.

Purpose

The Display MQM Object Names (DSPMQMOBJN) command is used to provide the AS/400 name and type for a specified MQM object name and type, or to display the MQM name and type for a specified AS/400 object name and type.

The directories searched are QMQMDATA and QMQMPROC. The following details are displayed for the specified objects:

- The AS/400 object name
- The AS/400 object type
- The name of the library
- The MQM object type
- The 48-character MQM object name

Note: Chained user spaces for objects are not shown. Only the first object name in any chain is shown.

Required parameters

OBJ

Specifies the name of the objects for which the corresponding name and type is to be displayed. If OBJTYPE is specified as *ALL, *USRSPC or *USRIDX, this is a 10-character AS/400 object or generic object name, otherwise it is a 48-character MQM object or generic object name.

***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*, and 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.

OBJTYPE

Specifies the object type of the objects to be displayed. The type specified also indicates whether the OBJ parameter is a 10-character AS/400 name or a 48-character MQM object name.

<u>*ALLMQM</u>: All MQM objects with names specified by OBJ.

***ALL**: All AS/400 Objects of types *USRSPC, or *USRIDX with names specified by OBJ, that correspond to MQ objects.

***Q**: All MQM queues with names specified by OBJ.

***ALSQ**: All MQM alias queues with names specified by OBJ.

***LCLQ**: All MQM local queues with names specified by OBJ.

***RMTQ**: All MQM remote queues with names specified by OBJ.

***MDLQ**: Template used when a dynamic queue is created.

***PRC**: All MQM process definitions with names specified by OBJ.

***MQM**: The Message Queue Manager object with name specified by OBJ.

***ADM**: The MQM queue manager administration object with name specified by OBJ. This has the same name as the queue manager object.

***CTLG**: The MQM queue manager catalog object with name specified by OBJ. This has the same name as the queue manager object.

***USRSPC**: All AS/400 user space objects with name specified by OBJ.

***USRIDX**: All AS/400 user index objects with name specified by OBJ.

Optional parameters

OUTPUT

Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

<u>*</u>: 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.

DSPMQMPRC (Display MQM Process) Command

```
                                                         Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                         ┌─OUTPUT(*)──────┐
►►──DSPMQMPRC──PRCNAME(process-name)─────┤                ├(P)──────────────────────────────────────►◄
                                         └─OUTPUT(*PRINT)─┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

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

Required parameters

PRCNAME
The name of the process definition to be displayed.

process-name: Specify the name of the process definition.

Optional parameters

OUTPUT
Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

*****: 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.

DSPMQMQ (Display MQM Queue) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
                                 ┌─OUTPUT(*)──────┐
►►──DSPMQMQ──QNAME(queue-name)───┤                ├(P)──────────────────────────────────►◄
                                 └─OUTPUT(*PRINT)─┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Display MQM Queue (DSPMQMQ) command displays the attributes of an existing MQM queue definition.

Required parameters

QNAME
Specifies the name of the queue.

queue-name: Specify the name of the queue.

Optional parameters

OUTPUT
Specifies whether the output from the command is shown at the requesting workstation, or printed with the job's spooled output.

*****: Output requested by an interactive job is shown on the display. Output requested by a batch job is printed with the job's spooled output.

***PRINT**: The output is printed with the job's spooled output.

ENDMQM (End Message Queue Manager) Command

```
                                                    Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                           ┌─OPTION(*CNTRLD)─┐
►►──ENDMQM──MQMNAME(queue-manager-name)────┤                 ├(P)──────────────────────────────►◄
                                           └─OPTION(*IMMED)──┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The End Message Queue Manager (ENDMQM) command ends the local message queue manager. The attributes of the message queue manager are not affected and it can be restarted using the Start Message Queue Manager (STRMQM) command.

Required parameters

MQMNAME

Specifies the name of the message queue manager.

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

Optional parameters

OPTION

Specifies whether or not programs currently being processed are allowed to complete.

***CNTRLD**: Allow programs currently being processed to complete. An MQCONN call (or an MQOPEN or MQPUT1 which would perform an implicit connection) fails.

***IMMED**: End programs currently being processed, including utilities, 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.

In OS/400 V4R2 and later release the system behavior for activation group termination has changed. The system now ends the process, unless the activation group has been explicitly added to the program stack.

If you issue an ENDMQM *IMMED command, when a program is running that is bound to the AMQZSTUB or AMQVSTUB service programs, the following occurs:

- If the program has been called from a QCMD prompt your program is ended and you are logged off the system.
- If the program has been called from within an MQSeries panel, for example WRKMQMCHL, the program is ended but you are not logged off the system.

ENDMQMCHL (End MQM Channel) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
                                      ┌─OPTION(*CNTRLD)─┐
►►──ENDMQMCHL──CHLNAME(channel-name)──┤                 ├(P)──────────────────────────────────►◄
                                      └─OPTION(*IMMED)──┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The End MQM Channel (ENDMQMCHL) command closes an MQM channel, and the channel is no longer enabled for automatic restarts.

Restriction

This command is shipped with public *EXCLUDE authority and with the QPGMR and QSYSOPR user profiles having private authority to use the command.

Required parameters

CHLNAME

Specifies the name of the channel.

channel-name: Specify the name of the channel.

Optional parameters

OPTION

Specifies whether or not processing for the current batch of messages is allowed to finish in a controlled manner.

***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.

ENDMQMCSVR (End MQM Command Server) Command

```
                                             Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
                                            ┌─OPTION(*CNTRLD)─┐
►►──ENDMQMCSVR──MQMNAME(queue-manager-name)─┤                 ├─(P)──────────────────────►◄
                                            └─OPTION(*IMMED)──┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The End MQM Command Server (ENDMQMCSVR) command stops the MQM command server for the specified local queue manager.

Required parameters

MQMNAME

Specifies the name of the local queue manager for which the command server is to be ended.

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

Optional parameters

OPTION

Specifies whether or not the command message currently being processed is allowed to complete.

***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 may not be completed.

ENDMQMSRV (End MQM Service Job) Command

```
                                                Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──ENDMQMSRV──────────────────────────────────────────────────────────────────────────────►◄
```

Purpose

The End MQM Service Job (ENDMQMSRV) command ends the remote job service operation started by a Start MQM Service Job (STRMQMSRV) command.

Restriction

To use this command, you must have *ALLOBJ authority, or be signed on to the system with one of the following user IDs:

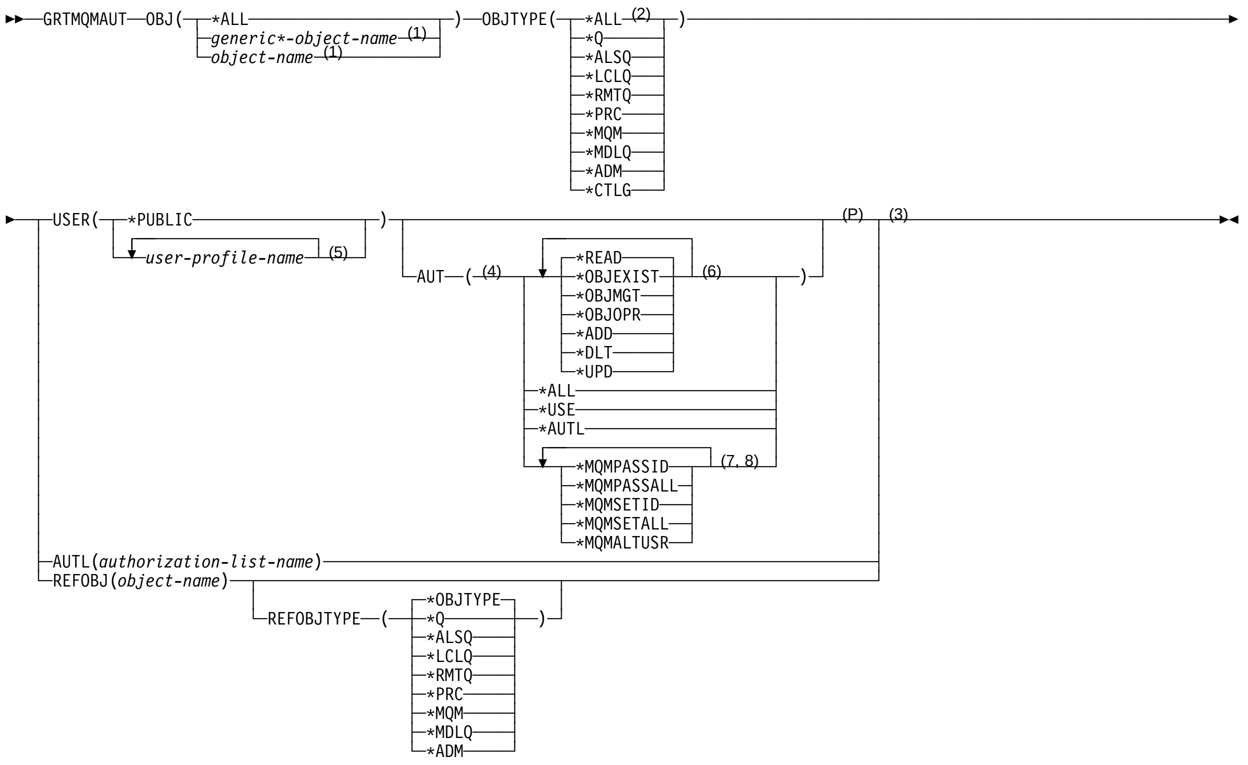
- QPGMR
- QSYSOPR
- QSRV
- QSRVBAS

Required parameters

There are no parameters for this command.

GRTMQMAUT (Grant MQM Authority) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec



Notes:

1 Object name and generic object names are 48-character MQM queue-manager, queue or process names.

2 OBJTYPE(*ALL) is not valid if REFOBJTYPE(*OBJTYPE) is specified.

3 Positional order is USER, AUT, AUTL, REFOBJ, REFOBJTYPE. Only one of USER, AUTL, or REFOBJ can be specified.

4 AUT(*AUTL) is valid only if USER(*PUBLIC) is specified.

5 A maximum of 50 repetitions.

6 Each can be used only once, a maximum of 7.

7 Valid for OBJTYPE(*ADM) only.

8 Each can be used only once, a maximum of 5.

P All parameters preceding this point can be specified positionally.

Purpose

The Grant MQM Authority (GRTMQMAUT) command is used to grant specific authority for the object 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, either for the object or for the authorization list
- Groups of users who do not have any authority to the object, or are not on the authorization list that secures the object
- Users of the referenced object as specified in the REFOBJ parameter
- Users on an established authorization list

The GRTMQMAUT command can be used by an object's owner, by the security officer, or by a user with object management authority for the specified object. Users with object management authority can grant any authority that they have, excluding object management, to other users. Only the owner of the object or someone with all-object (*ALLOBJ) special authority can grant object management authority to a user.

A user with *ALL authority can assign a new authorization list.

Required parameters

OBJ

Specifies the name or names of the MQM objects for which authorities are being granted.

***ALL**: All objects of the type specified by the value of the OBJTYPE parameter.

object-name: Specify the name of an MQM object for which specific authority is given to one or more users.

generic-object-name*: Specify the generic name of the objects to be selected. A generic name is a character string followed by an asterisk (*), for example ABC*, and the specified authorities are granted for all objects whose names 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.

OBJTYPE

Specifies the object type of the object for which specific authorities are being granted.

***ALL**: All MQM object types.

***Q**: All queue object types.

***ALSQ**: Alias queue.

***LCLQ**: Local queue.

***RMTQ**: Remote queue.

***PRC**: Process definition.

***MQM**: Message queue manager. This object holds the attributes of the message queue manager.

***MDLQ**: Template used when a dynamic queue is created.

***ADM**: Message queue manager administration object. The administration object has the same name as the message queue manager object. It holds the MQM authorities to specify message context, and to open MQM objects as an alternate user.

***CTLG**: Message queue manager catalog object. The catalog object has the same name as the message queue manager object. It holds the names of MQM objects. A user needs authorities on this object to be able to start or stop the message queue manager, or to create or delete MQM queues and process definitions.

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 MQM Authority (RVKMQMAUT) command.

***PUBLIC**: All users of the system, who do not have authority specifically given to them for the object, who are not on the authorization list, whose user group does not have any authority, or whose user group is not on the authorization list, are authorized to use the object as specified in the AUT parameter.

user-profile-name: Specify the names of one or more users who are to be granted specific authority for the object. Up to 50 user profile names can be specified.

AUT

Specifies the authority being given to the named users. Values for AUT can be specified in any one of three ways:

- As a set of up to seven specific object and data authorities
- As a single general authority
- For OBJTYPE(*ADM) only, as a set of up to five MQM authorities

Specify up to seven of the following:

***READ**: Read authority provides the authority needed to get the contents of an entry in an object. This is valid for message queue manager, queue, and process objects:

For any of these objects *READ provides the authority to open the object for inquire and to display an object using an MQM display command.

For queue and process objects, *READ also provides the authority to use the object as the source of a copy operation using an MQM copy command.

For a queue object, *READ also provides the authority to open a queue for browse or, together with *DLT, to open a queue for input.

For an administration object, *READ is used by the *MQMPASSID and *MQMPASSALL definitions.

***OBJEXIST**: Object existence authority provides the authority to control the object's existence and ownership. These rights are necessary for users who want to delete the object, perform save and restore operations for the object, or transfer ownership of the object. Note that a user who has special save system authority (*SAVSYS) does not need *OBJEXIST authority. Object existence authority is required to create an object that has been named by an authority holder. It can be used for queue, process, and catalog objects:

For queue and process objects, *OBJEXIST provides the authority to delete the object using an MQM delete command.

For a catalog object, *OBJEXIST provides the authority to delete the entire message queue manager using the MQM delete command.

***OBJMGT**: Object management authority provides the authority to specify the security for the object, move or rename the object.

***OBJOPR**: Object operational authority provides the authority to look at the description of an object, and use the object as determined by the data authorities that the user has to the object. It is valid for MQM objects of all types:

> For message queue manager, queue, and process objects, *OBJOPR, together with *UPD, provides the authority to change or replace an object with an MQM change, create, or copy command.
>
> For queue objects, *OBJOPR, together with the appropriate administration object authority, provides the authority to open a queue object with any of the pass context or set context options.
>
> For the catalog object, *OBJOPR provides authority to start and stop the message queue manager.
>
> For the administration object, *OBJOPR is used by the *MQMPASSALL definition.

***ADD**: Add authority provides the authority needed to add entries to an object. It is valid for queue, catalog, and administration objects:

> For a queue object, *ADD provides the authority to open the queue for output.
>
> For a catalog object, *ADD provides the authority to create a new message queue manager object, using an MQM create or copy command.
>
> For an administration object, *ADD is used by the *MQMSETALL definition.

***DLT**: Delete authority provides the authority needed to remove entries from an object. This is valid for queue, catalog, and administration objects:

> For a queue object, *DLT provides the authority to delete messages from a queue using the MQM delete command and, together with *READ authority, to open a queue for input.
>
> For a catalog object, *DLT provides the authority to delete queue and process objects using an MQM delete command.
>
> For the administration object, *DLT is used by the *MQMALTUSR definition.

***UPD**: Update authority provides the authority needed to change the entries in an object. This is valid for message queue manager, queue, process, and administration objects:

> For message queue manager, queue, and process objects, *UPD provides the authority to open an object for set and to change an object using an MQM change command.
>
> For an administration object, *UPD is used by the *MQMSETID and *MQMSETALL definitions.

Or specify one of the following:

***ALL**: All authority allows the user to perform all operations on the object, except those limited to the owner, or controlled by authorization list management authority. In addition to the functions allowed with *USE and

*CHANGE, the user can control the object's existence, and specify the security for the object.

***USE**: Use authority allows the user to perform basic functions on the object. The user is prevented from changing the object. Use authority provides object operational authority and read authority.

***AUTL**: The public authority of the authorization list securing the object is used. It is only allowed with USER(*PUBLIC).

Or, for OBJTYPE(*ADM) only, specify up to five of the following:

***MQMPASSID**: This provides the authority to open a queue to pass identity context data when putting a message; the user must have *OBJOPR authority for the queue. *MQMPASSID authority for *ADM object types is equivalent to *READ authority on other object types.

***MQMPASSALL**: This provides the authority to open a queue to pass all context data when putting a message; the user must have *OBJOPR authority for the queue. *MQMPASSALL authority for *ADM object types is equivalent to *READ + *OBJOPR authority on other object types.

***MQMSETID**: This provides the authority to open a queue to set identity context when putting a message; the user must have *OBJOPR authority for the queue. *MQMSETID authority for *ADM object types is equivalent to *UPD authority on other object types.

***MQMSETALL**: This provides the authority to open a queue to set all context when putting a message and also to open a transmission queue directly for output; the user must have *OBJOPR and *ADD authorities, respectively, for the queue. *MQMSETALL authority for *ADM object types is equivalent to *UPD + *ADD authority on other object types.

***MQMALTUSR**: This provides the authority to specify an alternate user identity when opening an object. *MQMALTUSR authority for *ADM object types is equivalent to *DLT authority on other object types.

AUTL
Specifies the name of the authorization list whose members are given authority for the object specified in the OBJ parameter.

authorization-list-name: Specify the name of the authorization list.

REFOBJ
Specifies the name of the object being queried to obtain authorization information; these authorizations are then given to the object specified by the OBJ parameter. Users authorized to the referenced object are authorized in the same manner to the object for which authority is being given. If the referenced object is secured by an authorization list, that authorization list secures the object specified in the OBJ parameter.

object-name: Specify the name of the object.

REFOBJTYPE
Specifies the object type of the referenced object.

<u>***OBJTYPE**</u>: The object type of the referenced object is the same as that for the object specified by the OBJTYPE parameter.

***Q**: All queue object types.

***ALSQ**: Alias queue.

***LCLQ**: Local queue.

***RMTQ**: Remote queue.

***PRC**: Process definition.

***MQM**: Message queue manager.

***MDLQ**: Template used when a dynamic queue is created.

***ADM**: Message queue manager administration object.

PNGMQMCHL (Ping MQM Channel) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
                                  ┌─DATACNT(64)─────────┐     ┌─CNT(1)──────────┐
►►──PNGMQMCHL──CHLNAME(channel-name)──┼─────────────────────┼─(P)─┼─────────────────┼────────────────────────►◄
                                  └─DATACNT(data-count)─┘     └─CNT(ping-count)─┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Ping MQM Channel (PNGMQMCHL) command tests a channel by sending data, as a special message, to the remote message queue manager and checking that the data is returned. This command only works from the channel end sending the message, and the data is generated by the local message queue manager.

Required parameters

CHLNAME

Specifies the name of the channel.

channel-name: Specify the name of the channel.

Optional parameters

DATACNT

Specifies the length of the data.

64: The default value is 64 bytes.

data-count: Specify a value in the range 16 through 32 768.

CNT

Specifies the number of times that the channel is to be pinged.

1: The channel is pinged once.

ping-count: Specify a value in the range 1 through 16.

RCDMQMIMG (Record MQM Object Image) Command

```
                                              Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──RCDMQMIMG──OBJ(──┬─*ALL──────────────────┬─)──OBJTYPE(──┬─*ALL───┬─)(P)──────────────────────────────►◄
                     ├─generic*-object-name──┤              ├─*Q─────┤
                     └─object-name───────────┘              ├─*ALSQ──┤
                                                            ├─*LCLQ──┤
                                                            ├─*RMTQ──┤
                                                            ├─*PRC───┤
                                                            ├─*MQM───┤
                                                            ├─*MDLQ──┤
                                                            ├─*ADM───┤
                                                            └─*CTLG──┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Record MQM Object Image (RCDMQMIMG) command is used to provide a marker for the selected set of MQM 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 MQM Object (RCRMQMOBJ) Command, on this set of MQM objects, to succeed.

Restriction

This command is shipped with public *EXCLUDE authority and with the QPGMR and QSYSOPR user profiles having private authority to use the command.

Required parameters

OBJ

Specifies the name of the objects which should be recorded. This is a 48-character MQM object or generic object name.

***ALL**: All MQM 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*, and 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 MQM object to be recorded.

OBJTYPE

Specifies the object type of the objects to be re-created.

***ALL**: Specifies all MQM object types.

***Q**: Specifies MQM queue objects with names specified by OBJ.

***ALSQ**: Specifies MQM alias queue objects with names specified by OBJ.

***LCLQ**: Specifies MQM local queue objects with names specified by OBJ.

***RMTQ**: Specifies MQM remote queue objects with names specified by OBJ.

***PRC**: Specifies MQM process objects with names specified by OBJ.

***MQM**: The Message Queue Manager object with name specified by OBJ.

***MDLQ**: Template used when a dynamic queue is created.

***ADM**: The MQM queue manager administration object with name specified by OBJ. This has the same name as the queue manager object.

***CTLG**: The MQM queue manager catalog object with name specified by OBJ. This has the same name as the queue manager object.

RCRMQMOBJ (Recreate MQM Object) Command

```
                                                          Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──RCRMQMOBJ──OBJ(──┬─*ALL───────────────────┬─)──OBJTYPE(──┬─*ALL──┬─)─(P)──────────────────────────►◄
                     ├─generic*-object-name─┤                ├─*Q────┤
                     └─object-name──────────┘                ├─*ALSQ─┤
                                                             ├─*LCLQ─┤
                                                             ├─*MDLQ─┤
                                                             ├─*CTLG─┤
                                                             ├─*MQM──┤
                                                             ├─*ADM──┤
                                                             ├─*RMTQ─┤
                                                             └─*PRC──┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Recreate MQM Object (RCRMQMOBJ) command is used to provide a recovery mechanism for damaged MQM objects. The command completely re-creates the object from information recorded in the MQM journals. If no damaged objects exist, no action is performed.

Restriction

This command is shipped with public *EXCLUDE authority and with the QPGMR and QSYSOPR user profiles having private authority to use the command.

Required parameters

OBJ

Specifies the name of the objects which should be re-created if they are damaged. This is a 48-character MQM object or generic object name.

***ALL**: All damaged MQM objects of the specified type (OBJTYPE) are re-created.

generic-object-name*: Specify the generic name of the objects to be recovered. A generic name is a character string followed by an asterisk (*), for example ABC*, and 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 MQM object which is to be re-created if it is damaged.

OBJTYPE

Specifies the object type of the objects to be re-created.

***ALL**: Specifies all MQM object types.

***Q**: Specifies all MQM queue objects with names specified by OBJ.

***ALSQ**: Specifies all MQM alias queue objects with names specified by OBJ.

***LCLQ**: Specifies all MQM local queue objects with names specified by OBJ.

***MDLQ**: Template used when a dynamic queue is created.

***CTLG**: Message queue manager catalog object. The catalog object has the same name as the message queue manager object. It holds the names of MQM objects. A user needs authorities on this object to be able to start or stop the message queue manager, or, to create or delete MQM queues and process definitions.

***MQM**: Message queue manager. This object holds the attributes of the message queue manager.

***ADM**: Message queue manager administration object. The administration object has the same name as the message queue manager object. It holds the MQM authorities to specify message context, and to open MQM objects as an alternate user.

***RMTQ**: Specifies all MQM remote queue objects with names specified by OBJ.

***PRC**: Specifies all MQM process objects with names specified by OBJ.

RSTMQMCHL (Reset MQM Channel) Command

```
                                                        Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
                                          ┌─MSGSEQNUM(1)──────────────────────┐
►►──RSTMQMCHL──CHLNAME(channel-name)──────┼───────────────────────────────────┼(P)───────────────►◄
                                          └─MSGSEQNUM(message-sequence-number)┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Reset MQM Channel (RSTMQMCHL) command resets the message sequence number for an MQM channel to a specified sequence number for use the next time that the channel is started.

This command can only be used for sender (*SDR) and server (*SVR) channels.

Restriction

This command is shipped with public *EXCLUDE authority and with the QPGMR and QSYSOPR user profiles having private authority to use the command.

Required parameters

CHLNAME

Specifies the name of the channel.

channel-name: Specify the name of the channel.

Optional parameters

MSGSEQNUM

Specifies the new message sequence number.

1: The new message sequence number is 1.

message-sequence-number: Specify the new message sequence number. It must be in the range 1 through 999 999 999.

RSVMQMCHL (Resolve MQM Channel) Command

```
                                                   Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──RSVMQMCHL──CHLNAME(channel-name)──OPTION(──┬─*CMT─┬──)(P)──────────────────────────────────►◄
                                               └─*BCK─┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Resolve MQM 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, as to whether or not the messages were received. Any outstanding units of work need to be resolved with either backout or commit.

This command can only be used for sender (*SDR) and server (*SVR) channels.

Restriction

This command is shipped with public *EXCLUDE authority and with the QPGMR and QSYSOPR user profiles having private authority to use the command.

Required parameters

CHLNAME
Specifies the name of the channel.

channel-name: Specify the name of the channel.

OPTION
Specifies whether to back out or commit the messages.

***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.

RVKMQMAUT (Revoke MQM Authority) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
►►──RVKMQMAUT──OBJ(──┬─*ALL──────────────────────┬─)──OBJTYPE(──┬─*ALL──┬─)──────────────────────────────────►
                     ├─generic*-object-name─(1)──┤              ├─*Q────┤
                     └─object-name─(1)───────────┘              ├─*ALSQ─┤
                                                                ├─*LCLQ─┤
                                                                ├─*RMTQ─┤
                                                                ├─*PRC──┤
                                                                ├─*MDLQ─┤
                                                                ├─*MQM──┤
                                                                ├─*ADM──┤
                                                                └─*CTLG─┘

                                                  ┌────────────────┐
►──┬─USER(──┬─*ALL─────────────────────┬─)AUT(──┬──▼─┬─*READ─────┬─┴─(4)───┬─)(2)─┬─(P)──────────────────────►◄
   │        ├─*PUBLIC──────────────────┤        │    ├─*OBJEXIST─┤         │      │
   │        │  ┌───────────────────┐   │        │    ├─*OBJMGT───┤         │      │
   │        └──▼─user-profile-name─┴(3)┘        │    ├─*OBJOPR───┤         │      │
   │                                            │    ├─*ADD──────┤         │      │
   │                                            │    ├─*DLT──────┤         │      │
   │                                            │    └─*UPD──────┘         │      │
   │                                            ├─*ALL─────────────────────┤      │
   │                                            ├─*USE─────────────────────┤      │
   │                                            ├─*AUTL────────────────────┤      │
   │                                            │  ┌───────────────────┐   │      │
   │                                            └──▼─┬─*MQMPASSID───┬───┴(5)┘      │
   │                                                 ├─*MQMPASSALL──┤              │
   │                                                 ├─*MQMSETID────┤              │
   │                                                 ├─*MQMSETALL───┤              │
   │                                                 └─*MQMALTUSR───┘              │
   └─AUTL(authorization-list-name)─────────────────────────────────────────────────┘
```

Notes:

1 Object name and generic object name represent 48 character MQM queue, process and storage class objects names.

2 AUT(*AUTL) is valid only if USER(*PUBLIC) is specified.

3 A maximum of 50 repetitions.

4 Select one or more, a maximum of 7.

5 Valid for OBJTYPE(*ADM) only. Each can be used only once, a maximum of 5.

P All parameters preceding this point can be specified positionally.

Purpose

The Revoke MQM Authority (RVKMQMAUT) command is used to take away specific or all authority for named objects, from one or more users also named in the command, or to remove the authority of an authorization list for the named objects.

This command can be used by an object's owner, by the security officer, or by a user with change authority for the object having its authority removed. A user who has change authority can remove only the authorities specific to that user. A user may not be able to give or remove authorities for an object that has been allocated (locked) to another job. If a specific (not *ALL) authority cannot be revoked, a message is issued identifying the authorities that were not revoked.

Security risk

Revoking all authorities specifically given to a user for an object can result in the user having more authority than before the revoke operation. If a user has *USE authority for an object, and *CHANGE authority on the authorization list that secures the object, revoking *USE authority results in the user having *CHANGE authority to the object.

Required parameters

OBJ

Specifies the name of the objects for which specific authority is revoked.

***ALL**: All objects of the specified type (OBJTYPE).

generic-object-name*: Specify the generic name of the objects. A generic name is a character string followed by an asterisk (*), for example, ABC*. If a generic name is specified, then specific authorities are revoked for all objects with names that begin with the generic name, and for which the user has authority.

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: Specify the name of the object for which specific authorities are revoked.

OBJTYPE

Specifies the object type of the objects for which specific authorities are revoked.

***ALL**: All MQM object types.

***Q**: All queue object types.

***ALSQ**: Alias queue.

***LCLQ**: Local queue.

***RMTQ**: Remote queue.

***PRC**: Process definition.

***MDLQ**: Template used when a dynamic queue is created.

***MQM**: Message queue manager. This object holds the attributes of the message queue manager.

***ADM**: Message queue manager administration object. The administration object has the same name as the message queue manager object. It holds the MQM authorities to specify message context, and to open MQM objects as an alternate user.

***CTLG**: Message queue manager catalog object. The catalog object has the same name as the message queue manager object. It holds the names of MQM objects. A user needs authorities on this object to be able to start or stop the message queue manager, or, to create or delete MQM queues and process definitions.

Optional parameters

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 MQM 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.

***ALL**: The authorities specified in the AUT parameter are taken away from all enrolled users of the system except the owner, if they are publicly or explicitly authorized.

***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. A maximum of 50 user profile names can be specified.

AUT

Specifies the authority being taken away from the users specified in the USER parameter. Values for AUT can be specified in any one of three ways: as a set of up to seven specific object and data authorities, as a single general authority, or, for OBJTYP(*ADM) only, as a set of up to five MQM authorities.

Specify up to seven of the following:

***READ**: Read authority provides the authority needed to get the contents of an entry in an object. This is valid for message queue manager, queue, process, and administration objects:

For the message queue manager, queue, and process objects, *READ provides the authority to open an object for inquire and to display an object using an MQM display command.

For queue and process objects, *READ provides the authority to use the object as the source of a copy operation using an MQM copy command.

For a queue object, *READ also provides the authority to open a queue for browse or, together with *DLT, to open a queue for input.

For an administration object, *READ is used by the *MQMPASSID and *MQMPASSALL definitions.

***OBJEXIST**: Object existence authority provides the authority to control the object's existence and ownership. These rights are necessary for users who want to delete the object, perform save and restore operations for the object, or transfer ownership of an object. If a user has special save system authority, *SAVSYS, that user does not need *OBJEXIST authority. Object existence authority is required to create an object that has been named by an authority holder. This is valid for the queue, process, and catalog objects:

For queue and process objects, *OBJEXIST provides the authority to delete the object using an MQM delete command.

For a catalog object, *OBJEXIST provides the authority to delete the entire message queue manager using the MQM delete command.

***OBJMGT**: Object management authority provides the authority to specify the security for the object, and to move or rename the object.

***OBJOPR**: Object operational authority provides the authority to look at the description of an object, and to use the object as determined by the data authorities that the user has to the object. This is valid for all MQM objects:

For message queue manager, queue, and process objects, *OBJOPR, together with *UPD, provides the authority to change or replace an object with an MQM change, create, or copy command.

For queue objects, *OBJOPR, together with the appropriate administration object authority, provides the authority to open a queue object with any of the pass context or set context options.

For the catalog object, *OBJOPR provides authority to start and stop the message queue manager.

For the administration object, *OBJOPR is used by the *MQMPASSALL definition.

***ADD**: Add authority provides the authority needed to add entries to an object. This is valid for queue, catalog, and administration objects:

For a queue object, *ADD provides the authority to open the queue for output.

For a catalog object, *ADD provides the authority to create a new message queue manager object, using an MQM create or copy command.

For an administration object, *ADD is used by the *MQMSETALL definition.

***DLT**: Delete authority provides the authority needed to remove entries from an object. This is valid for queue, catalog, and administration objects:

For a queue object, *DLT provides the authority to delete messages from a queue using the MQM delete command and, together with *READ authority, to open a queue for input.

For a catalog object, *DLT provides the authority to delete queue and process objects using an MQM delete command.

For the administration object, *DLT is used by the *MQMALTUSR definition.

***UPD**: Update authority provides the authority needed to change the entries in an object. This is valid for message queue manager, queue, process, and administration objects:

For message queue manager, queue, and process objects, *UPD provides the authority to open an object for set and to change an object using an MQM change command.

For an administration object, *UPD is used by the *MQMSETID and *MQMSETALL definitions.

Or specify one of the following:

***ALL**: All authority allows the user to perform all operations on the object, except those limited to the owner, or controlled by authorization list management authority. In addition to the functions allowed with *USE and *CHANGE, the user can control the object's existence and specify the security for the object.

***USE**: Use authority allows the user to perform basic functions on the object. The user is prevented from changing the object. Use authority provides object operational authority and read authority.

***AUTL**: The public authority of the authorization list specified in the AUTL parameter is revoked for the object. The public authority for the object becomes *EXCLUDE. Specify the name of the authorization list whose authority is removed from the object.

Or, for OBJTYP(*ADM) only, specify up to five of the following:

***MQMPASSID**: This provides the authority to open a queue to pass identity context data when putting a message; the user must have *OBJOPR authority for the queue. *MQMPASSID authority for *ADM object types is equivalent to *READ authority on other object types.

***MQMPASSALL**: This provides the authority to open a queue to pass all context data when putting a message; the user must have *OBJOPR authority for the queue. *MQMPASSALL authority for *ADM object types is equivalent to *READ + *OBJOPR authority on other object types.

***MQMSETID**: This provides the authority to open a queue to set identity context when putting a message; the user must have *OBJOPR authority for the queue. *MQMSETID authority for *ADM object types is equivalent to *UPD authority on other object types.

***MQMSETALL**: This provides the authority to open a queue to set all context when putting a message and also to open a transmission queue directly for output; the user must have *OBJOPR and *ADD authorities, respectively, for the queue. *MQMSETALL authority for *ADM object types is equivalent to *UPD + *ADD authority on other object types.

***MQMALTUSR**: This provides the authority to specify an alternate user identity when opening an object. *MQMALTUSR authority for *ADM object types is equivalent to *DLT authority on other object types.

AUTL

Specifies the name of the authorization list that is revoked from the object specified in the OBJ parameter. If public authority in the object is *AUTL, it is changed to *EXCLUDE. The authorization list’s authority is then removed.

authorization-list-name: Specify the name of the authorization list.

STRMQM (Start Message Queue Manager) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
           ┌─MQMNAME(*BLANK)──────────────┐
►►──STRMQM─┤                              ├(P)──────────────────────────────────────────────►◄
           └─MQMNAME(queue-manager-name)──┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Start Message Queue Manager (STRMQM) command starts the local queue manager.

Optional parameters

MQMNAME
Specifies the name of the message queue manager.

For a remote request issued from the administration utility, this parameter specifies the name of the target message-queue manager on which the application is to be processed.

***BLANK**: The queue-manager name is not inserted.

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

STRMQMADM (Start MQM Administrator) Command

```
                                                                          Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──STRMQMADM──GRPNAME(─┬─*ALL───────┬─)──MQMNAME(─┬─*ALL─────────────────────────────┬─)(P)──STRMQM(─┬─*YES─┬─)──────►
                        └─group-name─┘             ├─queue-manger-name────────────────┤               └─*NO──┘
                                                   └─generic*-queue-manager-name──────┘
►──STRCSVR(─┬─*YES─┬─)────────────────────────────────────────────────────────────────────────────────────────────────►◄
            └─*NO──┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Start MQM Administrator (STRMQMADM) command starts the MQM administration utility for the current workstation.

Required parameters

GRPNAME
Specifies the name of the group in which the queue manager is placed.

***ALL**: All groups for which this administrator has been assigned responsibility are selected.

group-name: Specify the name of the group to be selected. The maximum length of the character string is 10 bytes.

MQMNAME
Specifies the name or names of the queue managers.

***ALL**: All queue managers for which this administrator has been assigned responsibility are selected.

queue-manager-name: Specify the name of the queue manager to be selected. The maximum length of the string is 48 bytes. See “Names” on page 102.

generic-queue-manager-name*: Specify the generic name of the queue managers to be selected. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all queue managers 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.

STRMQM
Specifies whether or not the queue manager is to be started.

***YES**: The queue manager is started.

***NO**: The queue manager is not started.

STRCSVR
Specifies whether or not the command server is to be started.

***YES**: The command server is started.

***NO**: The command server is not started.

STRMQMCHL (Start MQM Channel) Command

```
                                              Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──STRMQMCHL──CHLNAME(channel-name)─(P)──────────────────────────────────────────────►◄
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Start MQM Channel (STRMQMCHL) command starts an MQM channel.

If the channel fails to start successfully, issue a Work with Spooled Files (WRKSPLF) command, and select the option to display data in the spooled file for user QMQM, to determine the cause of the failure.

Restriction

This command is shipped with public *EXCLUDE authority and with the QPGMR and QSYSOPR user profiles having private authority to use the command.

Required parameters

CHLNAME
Specifies the name of the channel definition.

channel-name: Specify the name of the channel.

STRMQMCHLI (Start MQM Channel Initiator) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
►►──STRMQMCHLI──QNAME(queue-name)─(P)──────────────────────────────────────────►◄
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Start MQM Channel Initiator (STRMQMCHLI) command starts an MQM channel initiator.

Restriction

This command is shipped with public *EXCLUDE authority and with the QPGMR and QSYSOPR user profiles having private authority to use the command.

Required parameters

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.

queue-name: Specify the name of the queue.

STRMQMCSVR (Start MQM Command Server) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
►►──STRMQMCSVR──MQMNAME(queue-manager-name)─(P)──────────────────────────────►◄
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Start MQM Command Server (STRMQMCSVR) command starts the MQM command server for the queue manager.

Required parameters

MQMNAME
Specifies the name of the queue manager.

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

STRMQMDLQ (Start MQSeries Dead-Letter Queue Handler)

```
                                                          Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──STRMQMDLQ──UDLMSGQ(──┬─*DFT───────────────────────────┬─)──SRCMBR(──┬─*FIRST──────────────┬─)(P)──────────────────►
                         ├─undelivered-message-queue-name─┤             └─source-member-name──┘
                         └─' '────────────────────────────┘

     ┌─SRCFILE(*LIBL/QTXTSRC)──────────────────────────────┐
►────┼─────────────────────────────────────────────────────┼──────────────────────────────────────────────────────►◄
     └─SRCFILE(──┬─*LIBL────────────────┬─/─┬─QTXTSRC──────────┬─)─┘
                 ├─*CURLIB──────────────┤   └─source file name─┘
                 └─source-library-name──┘
Note:
P  All parameters preceding this point can be specified positionally.
```

Purpose

Use the Start MQSeries 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.

For more information about rules tables and how to construct them, see “The DLQ handler rules table” on page 88.

Required parameters

UDLMSGQ

Specifies the name of the local undelivered-message queue that is processed.

***DFT**: The local undelivered-message queue that is used, is taken from the default queue manager for the installation.

undelivered-message-queue-name: Specify the name of the local undelivered-message queue.

' ': The queue that is named in the INPUTQ value in the rules table, or the system-default dead-letter queue is used.

For further information see “Control data” on page 88.

SRCMBR

Specifies the name of the source member, containing the user-written rules table that is being processed.

***FIRST**: Use the first member of the file.

source-member-name: Specify the name of the source member.

Optional parameters

SRCFILE
Specifies the qualified name of the file and library, in the form LIBRARY/FILE, containing the user-written rules table that is being processed.

***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.

QTXTSRC: Use QTXTSRC.

source-file-name: Specify the name of the source file.

STRMQMLSR (Start MQM Listener) Command

```
                                                  Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──STRMQMLSR─────────────────────────────────────────────────────────────────────────────────►◄
```

Purpose

The Start MQM Listener (STRMQMLSR) command starts an MQM listener.

This command is only valid for TCP/IP transmission protocols.

Restriction

This command is shipped with public *EXCLUDE authority and with the QPGMR and QSYSOPR user profiles having private authority to use the command.

Required parameters

There are no parameters to this command.

STRMQMSRV (Start MQM Service Job) Command

```
                                              Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──STRMQMSRV──JOB(──┬─job-name─────────────────────┬─)(P)──────────────────────────────►◄
                     ├─user-name/job-name───────────┤
                     └─job-number/user-name/job-name┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Start MQM Service Job (STRMQMSRV) command starts the remote service operation for a specified job so that a subsequent Trace MQM (TRCMQM) command can be issued to service that job.

Note that the specified job should be the job that issues the STRMQMSRV command.

The service operation continues until an End MQM Service Job (ENDMQMSRV) command is issued.

Restriction

To use this command, you must have `*ALLOBJ` authority, or be signed on to the system with one of the following user IDs:

- QPGMR
- QSYSOPR
- QSRV
- QSRVBAS

Required parameters

JOB

Specifies the name of the job being serviced. If no job number is given, all of the jobs currently in the system are searched for the specified job name. If duplicates of the specified name are found, a “select job” display is shown, and the user name and job number must be specified. The job name entered should not be the name of the job issuing the command.

A job identifier is a qualified name with up to three elements. For example:

job-name
user-name/job-name
job-number/user-name/job-name

job-name: Specify the name of the job being serviced.

user-name: Specify the name of the user of the job being serviced.

job-number: Specify the number of the job being serviced.

STRMQMMQSC (Start MQSeries Commands) Command

```
                                                                    Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                           ┌─SRCFILE(*LIBL/QMQSC)──────────────────────────────────┐
►►──STRMQMMQSC──SRCMBR(──┬─*FIRST─────────────┬─)(P)──┼───────────────────────────────────────────────────┼──────────────►
                         └─source-member-name─┘       │          ┌─*LIBL──────────────┐    ┌─QMQSC────────────┐   │
                                                      └─SRCFILE(─┼────────────────────┼─/──┼──────────────────┼─)─┘
                                                                 ├─*CURLIB────────────┤    └─source file name─┘
                                                                 └─source-library-name┘

►──┬──────────────────────────┬───────────────────────────────────────────────────────────────────────────────────────►◄
   │         ┌─*RUN────┐      │
   └─OPTION(─┼─────────┼─)────┘
             └─*VERIFY─┘

Note:
P   All parameters preceding this point can be specified positionally.
```

Purpose

The Start MQSeries Commands (STRMQMMQSC) command initiates a set of MQSeries 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.

Required parameters

SRCMBR

Specifies the name of the source member, containing the MQSC, to be processed.

***FIRST**: The first member of the file is used.

source-member-name: Specify the name of the source member.

Optional parameters

SRCFILE

Specifies the qualified name of the file, in the form LIBRARY/FILE, that contains the MQSC to be processed.

<u>*LIBL</u>: 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.

<u>QMQSC</u>: QMQSC is used.

source-file-name: Specify the name of the source file.

OPTION

Specifies how the MQSC is to be processed.

<u>*RUN</u>: The MQSeries commands are processed.

***VERIFY**: The MQSeries commands are verified and a report is written, but the commands are not run.

TRCMQM (Trace MQM Job) Command

```
                                                  Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                ┌─SET(*ON)──────────┐  ┌─TRCTYPE(*ALL)──────────────┐  ┌─TRCCOMP(*ALL)─────────┐
►►──TRCMQM──────┼───────────────────┼──┼────────────────────────────┼──┼───────────────────────┼──────────►
                └─SET(──┬─*OFF─┬─)──┘  │             ┌──────────┐   │  ├─TRCCOMP───────────────┤
                        └─*END─┘       └─TRCTYPE(──▼─┬─*API──┬──┴─)─┘  ├─(─────────────────────┤
                                                     ├─*COMP─┤         │ ┌─────────────────┐   │
                                                     ├─*FUNC─┤         ├─▼component-id-list┴──┤
                                                     ├─*MSG──┤         └─)─────────────────────┘
                                                     └─*COMM─┘

   ┌─MAXSTG(1024)─────────────────┐  ┌─TRCFULL(*WRAP)─────┐  ┌─OUTPUT(*PRINT)───┐
►──┼──────────────────────────────┼──┼────────────────────┼──┼──────────────────┼─(P)──────────────────►
   └─MAXSTG(maximum-K-bytes)──────┘  └─TRCFULL(*STOPTRC)──┘  └─OUTPUT(*OUTFILE)─┘

   ┌─OUTFILE(*LIBL/)────────────────────────────────────────┐  ┌─OUTMBR(*FIRST)────────┐
►──┼────────────────────────────────────────────────────────┼──┼───────────────────────┼────────────────►◄
   └─OUTFILE(──┬─*CURLIB/───────┬─)database-file-name───────┘  ├─OUTMBR(member-name────┤
               └─library-name/──┘                              │ ┌─*REPLACE─┐          │
                                                               └─┼─*ADD─────┼──────────┘
                                                                 └─)────────┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Trace MQM Job (TRCMQM) command controls tracing in the job for which the current job has issued a Start MQM Service Job (STRMQMSRV) command or, if no such command has been issued, in the current job.

TRCMQM, which sets tracing on or off, can trace message queue interface (MQI) functions, function flow, and MQSeries for AS/400 components together with any messages issued by MQSeries for AS/400. The trace records are stored in a trace storage area and, when the trace is ended, can be written to a spooled printer file, QSYSPRT, or to a database output file.

The trace output from the serviced job is returned to the servicing job when trace is set off, or the serviced job ends.

Restrictions

1. This command is shipped with public *EXCLUDE authority. To use this command, you must be signed on to the system with one of the following user IDs:
   - QPGMR
   - QRJE
   - QSRV
   - QSRVBAS
   - QSYSOPR
2. The number of trace records processed between the start and end of the trace must not exceed one million.

Optional parameters

SET

Specifies the collection of trace records.

***ON**: The collection of trace records is started.

***OFF**: The collection of trace records is stopped, and the trace records are written to the spooled printer file or output file.

***END**: The collection of trace records is stopped, and all existing trace records are deleted. No spooled printer file is created.

TRCTYPE

Specifies the type of trace data to store in the trace file.

***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 five parameters, but each option can only appear once.

***API**: Trace entry and exit at API level, including “main”.

***COMP**: Trace entry and exit at Component level.

***FUNC**: Trace entry and exit at Function level.

***MSG**: Messages issued by MQSeries for AS/400 are stored in the trace file. The data stored is the message identifier and CCSID only.

***COMM**: Trace of communications buffers.

TRCCOMP

Specifies the scope of tracing by component.

***ALL**: All components are traced.

component-id-list: A blank-separated list of component identifiers that control which components are to be traced. You can specify more than one component from the following list of component identifiers, but each option can only appear once.

***INS**: The Installation component is traced.

***CSV**: The Common Services component is traced.

***LQM**: The Local Queue Manager component is traced.

***AIC**: The Application Interface component is traced.

***LQK**: The Local Queue Manager kernel component is traced.

***DHD**: The Data Hardening component is traced.

***QMT**: The Queue Management component is traced.

***OCT**: The Object Catalog component is traced.

***TMT**: The Transaction Management component is traced.

***LMT**: The Log Management component is traced.

***OAM**: The Object Authority Manager is traced.

***DCV**: The Data Conversion component is traced.

***CSR**: The Command Server component is traced.

***RQP**: The Remote Queue Processor component is traced.

***RAS**: The Remote API Server component is traced.

***CMD**: The MQSeries for AS/400 commands are traced.

***COM**: The Communications component is traced.

***ADM**: The Administrator component is traced.

MAXSTG
Specifies the maximum size of storage to be used for the collected trace records.

1024: The default maximum is 1024 kilobytes.

maximum-K-bytes: Specify a value in the range 1 through 16 000.

TRCFULL
Specifies the action to take when the available storage is full.

***WRAP**: When the trace file is full, the trace wraps to the beginning. The oldest trace records are written over by new ones as they are collected.

Note: The trace file header is not destroyed by *WRAP.

***STOPTRC**: Tracing stops when the trace file is full of trace records, but the program continues processing.

OUTPUT
Specifies whether the output from the command is printed with the job's spooled output, or directed to a database file.

***PRINT**: The output is printed with the job's spooled output.

***OUTFILE**: The output is directed to the database file specified in the OUTFILE parameter.

OUTFILE
Specifies the qualified name of the database file to which the output of the command is directed. If the file does not exist, the system creates a database file in the specified library: the text for the new file is **“OUTFILE created by TRCMQM”**, and its public authority is *EXCLUDE.

The possible library values are:

***LIBL**: The library list is used to locate the database file.

***CURLIB**: The current library for the job is used to locate the database file. If there is no current library for the job, the QGPL library is used.

library-name: Specify the name of the library.

database-file-name: Specify the name of the database file.

OUTMBR
Specifies the member of the database file to which the output is directed and the operation to perform.

Element 1: Member to receive output

***FIRST**: The first member in the file receives the output. If the member does not exist, the system creates a member with the name specified in the OUTFILE parameter.

member-name: Specify the name of the file member that receives the output. If the member does not exist, the system creates a member with the specified name.

Element 2: Operation to perform on member

<u>*REPLACE</u>: If the member exists, the system clears it before adding the new records.

***ADD**: If the member exists, the system adds the new records to the end of the existing records.

WRKMQMCHL (Work with MQM Channels) Command

```
                                              Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                               ┌─CHLTYPE(*ALL)───────────┐
►►──WRKMQMCHL──CHLNAME(──┬─*ALL──────────────────┬─)──┤                         ├(P)──────────────────────►◄
                         ├─generic*-channel-name─┤    └─CHLTYPE(──┬─*SDR───┬─)─┘
                         └─channel-name──────────┘                ├─*SVR───┤
                                                                  ├─*RCVR──┤
                                                                  ├─*RQSTR─┤
                                                                  └─*SVRCN─┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Work with MQM 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.

Required parameters

CHLNAME

Specifies the name or names of the MQM channel definitions to be selected

***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*, and it selects all channel definitions having names that start with the character string.

Notes:

1. You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.
2. 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.

Optional parameters

CHLTYPE

Specifies the type of channel definitions that are to be displayed.

***ALL**: All types of channels

***SDR**: Sender channels

***SVR**: Server channels

***RCVR**: Receiver channels

***RQSTR**: Requester channels

***SVRCN**: Server connection channels

WRKMQMCHST (Work with MQM Channel Status) Command

Job: B,I Pgm: B,I Mod: B,I REXX: B,I Exec

```
                                                ┌─CONNAME(*ALL)──────────────────────────────┐
►►──WRKMQMCHST──CHLNAME(──┬─*ALL──────────────────┬──)──┤                                            ├──────────────►
                          ├─generic*-channel-name─┤     └─CONNAME(──┬─generic*-connection-name─┬──)─┘
                          └─channel-name──────────┘                 └─connection-name──────────┘

   ┌─TMQNAME(*ALL)──────────────────────────────────────────┐
►──┤                                                        ├(P)──────────────────────────────────────────────────►◄
   └─TMQNAME(──┬─generic*-transmission-queue-name─┬──)──────┘
               └─transmission-queue-name──────────┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Work with MQM Channel Status (WRKMQMCHST) command allows you to work with the status of one or more channel definitions.

Required parameters

CHLNAME

Specifies the name of the channel definition.

***ALL**: All the available channels are selected.

generic-channel-name*: Specify the generic name of the required channels. A generic name is a character string followed by an asterisk (*), for example ABC*, and it selects all objects having names that start with the selected character string.

Notes:

1. You are recommended to specify the name required within quotation marks. Using this format ensures that your selection is precisely what you entered.
2. 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 required channel.

Optional parameters

CONNAME

Specifies the name of the machine to be connected.

***ALL**: All the channels are selected.

generic-connection-name*: Specify the generic name of the required channels.

connection-name: Specify the name of the machine as required for 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.

TMQNAME
Specifies the name of the transmission queue.

***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.

WRKMQMMSG (Work with MQM Messages) Command

```
                                                       Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                     ┌─FIRST(1)────────────────┐  ┌─MAXMSG(48)──────────────┐
►►──WRKMQMMSG──QNAME(queue-name)(P)──┼─────────────────────────┼──┼─────────────────────────┼──────────────►
                                     └─FIRST(message-number)───┘  └─MAXMSG(count-value)─────┘

   ┌─MAXMSGLEN(1024)──────────┐
►──┼──────────────────────────┼─────────────────────────────────────────────────────────────────────────►◄
   └─MAXMSGLEN(length-value)──┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Work with MQM 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).

Required parameters

QNAME

Specifies the name of the local queue.

queue-name: Specify the name of the local queue.

Optional parameters

FIRST

Specifies the number of the first message to display.

1: The number of the first message to display is 1.

message-number: Specify the number of the first message to display in the range 1 through 640 000.

MAXMSG

Specifies the maximum number of messages to display.

48: Display a maximum of 48 messages.

count-value: Specify a value for the maximum number of messages to display in the range 1 through 640 000.

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.

1024: The size of the message data is 1024 bytes.

length-value: Specify a value for the maximum size of message data, in the range 128 through 999 999.

WRKMQMPRC (Work with MQM Processes) Command

```
                                                 Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec
►►──WRKMQMPRC──PRCNAME(──┬─*ALL─────────────────┬─)(P)──────────────────────────────────────►◄
                         ├─generic*-process-name─┤
                         └─process-name──────────┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Work with MQM 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 MQM process object.

Required parameters

PRCNAME

Specifies the name or names of the process definitions.

***ALL**: All process definitions are selected.

generic-process-name*: Specify the generic name of the MQM process definitions. A generic name is a character string followed by an asterisk (*), for example ABC*, and 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 MQM process definition.

WRKMQMQ (Work with MQM Queues) Command

```
                                                Job: B,I  Pgm: B,I  Mod: B,I REXX: B,I  Exec

                                            ┌─QTYPE(*ALL)──────────┐
►►──WRKMQMQ──QNAME(──┬─*ALL───────────────┬─)──┤                      ├(P)──────────────────────────►◄
                     ├─generic*-queue-name─┤    └─QTYPE(──┬─*ALS─┬─)─┘
                     └─queue-name──────────┘              ├─*LCL─┤
                                                          ├─*MDL─┤
                                                          └─*RMT─┘
```

Note:
P All parameters preceding this point can be specified positionally.

Purpose

The Work with MQM 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 MQM Queue object.

Required parameters

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.

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*, and 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.

Optional parameters

QTYPE

This parameter can be specified to limit the queues that are displayed to a particular type.

ALL: All queue types

***ALS**: Alias queues

***LCL**: Local queues

***MDL**: Model queues

***RMT**: Remote queues

Part 3. Appendixes

© Copyright IBM Corp. 1994, 1998

Appendix A. Sample resource definitions

This appendix contains sample AS/400 CL programs and the sample resource definition for setting up the default queues using the MQSeries commands (see “AMQSCOMA” on page 257).

AMQSDEF4

```
/********************************************************************/
/*                                                                  */
/* Program name: AMQSDEF4                                           */
/*                                                                  */
/* Description: Sample CL program defining MQM queues               */
/*              Can be run, with changes as needed, after           */
/*              starting the MQM                                    */
/*                                                                  */
/* Statement:     Licensed Materials - Property of IBM              */
/*                                                                  */
/*                5769-MQ2                                          */
/*                (C) Copyright IBM Corporation 1994, 1998.         */
/*                                                                  */
/* Status: Version 4 Release 2 Modification 1                       */
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Function:                                                        */
/*                                                                  */
/*                                                                  */
/*   AMQSDEF4 is a sample CL program to create or reset the         */
/*   MQI resource of the Message Queue Manager (MQM).               */
/*   It includes the following.                                     */
/*                                                                  */
/*      -- specification of the standard default attributes         */
/*                                                                  */
/*      -- definition of standard MQM queues                        */
/*                                                                  */
/*   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                                     */
/*                                                                  */
/*   AMQSDEF4 has no parameters.                                    */
/*                                                                  */
/********************************************************************/
            PGM
            DCL        VAR(&COPYRIGHT) TYPE(*CHAR) LEN(50) +
                         VALUE('5769-MQ2 (C) Copyright IBM +
                         Corporation 1994 1998')
            DCL        VAR(&COPYTEXT)  TYPE(*CHAR) LEN(50)
            DCL        VAR(&PGMNAME) TYPE(*CHAR) LEN(10) +
                         VALUE(AMQSDEF4)

/********************************************************************/
/*                                                                  */
/*   Monitor for Error Messages                                     */
/*                                                                  */
/********************************************************************/
            MONMSG     MSGID(CPF0000) EXEC(GOTO CMDLBL(ERROR))
```

```
/********************************************************************/
/*                                                                  */
/*   Force Usage of Copyright Statement                             */
/*                                                                  */
/********************************************************************/
            QSYS/CHGVAR VAR(&COPYTEXT) VALUE(&COPYRIGHT)

/********************************************************************/
/*       DEFAULT ALIAS QUEUE                                        */
/*                                                                  */
/*   Create the default alias queue.                                */
/*     REPLACE(*YES) allows this command to reset the attributes    */
/*     to known values.                                             */
/*                                                                  */
/*     Note: because it's a default queue, all attributes           */
/*     applicable to an alias queue are explicit, and all           */
/*     users have READ access.                                      */
/*                                                                  */
/********************************************************************/
            CRTMQMQ     QNAME('SYSTEM.DEFAULT.ALIAS.QUEUE')         +
                        QTYPE(*ALS)  REPLACE(*YES)                  +
                                                                    +
                        TEXT(*BLANK)   /* Queue description      */+
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        DFTPTY(0)      /* Default priority        */+
                        DFTMSGPST(*NO) /* Default non-persistent  */+
                        GETENBL(*YES)  /* Get operations allowed  */+
                        TGTQNAME(' ')  /* Base queue name         */

    /* Provide public read access to the defaults  */
            GRTMQMAUT  OBJ(SYSTEM.DEFAULT.ALIAS.QUEUE) OBJTYPE(*Q) +
                         USER(*PUBLIC)

/********************************************************************/
/*       DEFAULT LOCAL QUEUE                                        */
/*                                                                  */
/*   Create the default local queue.                                */
/*     REPLACE(*YES) allows this command to reset the attributes    */
/*     to known values.                                             */
/*                                                                  */
/*     Note: because it's a default queue, all attributes           */
/*     applicable to a local queue are explicit, and all            */
/*     users have READ access.                                      */
/*                                                                  */
/********************************************************************/
            CRTMQMQ     QNAME('SYSTEM.DEFAULT.LOCAL.QUEUE')         +
                        QTYPE(*LCL)  REPLACE(*YES)                  +
                                                                    +
                        TEXT(*BLANK)   /* Queue description      */+
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        DFTPTY(0)      /* Default priority        */+
                        DFTMSGPST(*NO) /* Default non-persistent  */+
                        GETENBL(*YES)  /* Get operations allowed  */+
                                                                    +
                        MAXDEPTH(5000) /* Maximum queue depth     */+
                        MAXMSGLEN(4194304)                          +
                                       /* Maximum message length  */+
                        SHARE(*YES)    /* Shareable               */+
                        DFTSHARE(*YES) /* Input open option       */+
                        MSGDLYSEQ(*PTY)/* Delivery by priority    */+
                        HDNBKTCNT(*NO) /* Backout not hardened    */+
                        USAGE(*NORMAL) /* Normal queue usage      */+
                                                                    +
                        TRGENBL(*NO)   /* Trigger control off     */+
                        HIGHTHLD(80)   /* High event threshold    */+
                        LOWTHLD(20)    /* Low event threshold     */+
                        FULLEVT(*YES)  /* Queue full event control*/+
                        HIGHEVT(*NO)   /* Queue high event control*/+
                        LOWEVT(*NO)    /* Queue low  event control*/+
                        SRVITV(999999999)                       +
                                       /* Service event interval  */+
                        SRVEVT(*NONE)  /* Service event control   */+
                        DISTLIST(*NO)  /* Distribution List cntrl */+
            /* next six defaults apply if trigger is made active  */+
                        TRGTYPE(*FIRST)/* Trigger on first message*/+
                        TRGDEPTH(1)    /* Trigger on first message*/+
                        TRGMSGPTY(0)   /* Trigger on any priority */+
                        TRGDATA(' ')   /* Trigger data            */+
                        PRCNAME(' ')   /* Process name if trigger */+
                        INITQNAME(' ') /* Initiation queue name   */+
                                                                     +
       /* other attributes can be tested, but MQM takes no action */+
                     RTNITV(999999999) /* Retention time (hours)  */+
                        BKTTHLD(0)     /* Backout threshold       */+
                        BKTQNAME(' ')  /* Backout requeue name    */

    /* Provide public read access to the defaults  */
            GRTMQMAUT  OBJ(SYSTEM.DEFAULT.LOCAL.QUEUE) OBJTYPE(*Q) +
                         USER(*PUBLIC)

/********************************************************************/
/*       DEFAULT MODEL QUEUE                                        */
/*                                                                  */
/*   Create the default model queue.                                */
/*     REPLACE(*YES) allows this command to reset the attributes    */
/*     to known values.                                             */
/*                                                                  */
/*     Note: because it's a default queue, all attributes           */
/*     applicable to a local queue are explicit, and all            */
/*     users have READ access.                                      */
/*                                                                  */
/********************************************************************/
            CRTMQMQ     QNAME('SYSTEM.DEFAULT.MODEL.QUEUE')         +
                        QTYPE(*MDL)  REPLACE(*YES)                  +
                                                                     +
                        TEXT(*BLANK)   /* Queue description       */+
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        DFTPTY(0)      /* Default priority        */+
                        DFTMSGPST(*NO) /* Default non-persistent  */+
                        GETENBL(*YES)  /* Get operations allowed  */+
                                                                     +
                        MAXDEPTH(5000) /* Maximum queue depth     */+
                        MAXMSGLEN(4194304)                           +
                                       /* Maximum message length  */+
                        SHARE(*YES)    /* Shareable               */+
                        DFTSHARE(*YES) /* Input open option       */+
                        MSGDLYSEQ(*PTY)/* Delivery by priority    */+
                        HDNBKTCNT(*NO) /* Backout not hardened    */+
                        USAGE(*NORMAL) /* Normal queue usage      */+
                        DFNTYPE(*TEMPDYN)                            +
                                       /* Temporary dynamic queue */+
                                                                     +
                        TRGENBL(*NO)   /* Trigger control off     */+
                        HIGHTHLD(80)   /* High event threshold    */+
                        LOWTHLD(20)    /* Low event threshold     */+
                        FULLEVT(*YES)  /* Queue full event control*/+
                        HIGHEVT(*NO)   /* Queue high event control*/+
                        LOWEVT(*NO)    /* Queue low  event control*/+
                        SRVITV(999999999)                        +
                                       /* Service event interval  */+
                        SRVEVT(*NONE)  /* Service event control   */+
                        DISTLIST(*NO)  /* Distribution List cntrl */+
            /* next six defaults apply if trigger is made active  */+
                        TRGTYPE(*FIRST)/* Trigger on first message*/+
                        TRGDEPTH(1)    /* Trigger on first message*/+
                        TRGMSGPTY(0)   /* Trigger on any priority */+
                        TRGDATA(' ')   /* Trigger data            */+
                        PRCNAME(' ')   /* Process name if trigger */+
                        INITQNAME(' ') /* Initiation queue name   */+
                                                                  +
       /* other attributes can be tested, but MQM takes no action */+
                     RTNITV(999999999) /* Retention time (hours)  */+
                        BKTTHLD(0)     /* Backout threshold       */+
                        BKTQNAME(' ')  /* Backout requeue name    */

    /* Provide public read access to the defaults  */
            GRTMQMAUT  OBJ(SYSTEM.DEFAULT.MODEL.QUEUE) OBJTYPE(*Q) +
                        USER(*PUBLIC)

/********************************************************************/
/*       DEFAULT REMOTE QUEUE                                       */
/*                                                                  */
/*   Create the default remote queue.                               */
/*     REPLACE(*YES) allows this command to reset the attributes    */
/*     to known values.                                             */
/*                                                                  */
/*     Note: because it's a default queue, all attributes           */
/*     applicable to a remote queue are explicit, and all           */
/*     users have READ access.                                      */
/*                                                                  */
/********************************************************************/
            CRTMQMQ     QNAME('SYSTEM.DEFAULT.REMOTE.QUEUE')       +
                        QTYPE(*RMT)  REPLACE(*YES)                 +
                                                                  +
                        TEXT(*BLANK)   /* Queue description       */+
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        DFTPTY(0)      /* Default priority        */+
                        DFTMSGPST(*NO) /* Default non-persistent  */+
                        TMQNAME(*NONE) /* no transmission queue   */+
                        RMTQNAME(' ')  /* Queue name              */+
                        RMTMQMNAME(' ') /* Name of remote QM      */

    /* Provide public read access to the defaults  */
            GRTMQMAUT  OBJ(SYSTEM.DEFAULT.REMOTE.QUEUE) OBJTYPE(*Q) +
                        USER(*PUBLIC)

/********************************************************************/
/*       DEFAULT MQI PROCESS OBJECT                                 */
/*                                                                  */
/*   Create the default MQI process object                          */
/*     REPLACE(*YES) allows this command to reset the attributes    */
/*     to known values.                                             */
/*                                                                  */
/*     Note: Because it's a default object, all attributes          */
/*     applicable to a process object are explicit, and all         */
/*     users have READ access.                                      */
/*                                                                  */
/********************************************************************/
            CRTMQMPRC   PRCNAME('SYSTEM.DEFAULT.PROCESS')          +
                        REPLACE(*YES)                              +
                                                                  +
                         TEXT(*BLANK)   /* Object description      */+
                         APPTYPE(*OS400)/* Application type        */+
                         APPID(' ')     /* Application identifier  */+
                         USRDATA(' ')   /* User data               */+
                         ENVDATA(' ')   /* Environment data        */

    /* Provide public read access to the defaults  */
            GRTMQMAUT  OBJ(SYSTEM.DEFAULT.PROCESS) OBJTYPE(*PRC) +
                         USER(*PUBLIC)

/*******************************************************************/
/*       DEFAULT CHANNEL DEFINITIONS                               */
/*                                                                 */
/*   Create the defaults for each of the channel types             */
/*     REPLACE(*YES) allows this command to reset the attributes   */
/*     to known values.                                            */
/*                                                                 */
/*******************************************************************/
    /* Defaults for a SENDER channel   */
            CRTMQMCHL  CHLNAME(SYSTEM.DEF.SENDER) CHLTYPE(*SDR) +
                         REPLACE(*YES) TRPTYPE(*TCP) TEXT(' ') +
                         CONNAME(' ') TMQNAME(' ') MCANAME(*NONE) +
                         BATCHSIZE(50)  DSCITV(6000) +
                         SHORTTMR(60)   SHORTRTY(10)  +
                         LONGTMR(1200)  LONGRTY(999999999) +
                         SCYEXIT(*NONE) SCYUSRDATA(*NONE)  +
                         SNDEXIT(*NONE) SNDUSRDATA(*NONE)  +
                         RCVEXIT(*NONE) RCVUSRDATA(*NONE)  +
                         MSGEXIT(*NONE) MSGUSRDATA(*NONE)  +
                         SEQNUMWRAP(999999999) +
                         MAXMSGLEN(4194304) CVTMSG(*NO) +
                         HRTBTINTVL(300) NPMSPEED(*FAST)

    /* Defaults for a SERVER channel   */
            CRTMQMCHL  CHLNAME(SYSTEM.DEF.SERVER) CHLTYPE(*SVR) +
                         REPLACE(*YES) TRPTYPE(*TCP) TEXT(' ') +
                         CONNAME(' ') TMQNAME(' ') MCANAME(*NONE) +
                         BATCHSIZE(50)  DSCITV(6000) +
                         SHORTTMR(60)   SHORTRTY(10)  +
                         LONGTMR(1200)  LONGRTY(999999999) +
                         SCYEXIT(*NONE) SCYUSRDATA(*NONE)  +
                         SNDEXIT(*NONE) SNDUSRDATA(*NONE)  +
                         RCVEXIT(*NONE) RCVUSRDATA(*NONE)  +
                         MSGEXIT(*NONE) MSGUSRDATA(*NONE)  +
                         SEQNUMWRAP(999999999) +
                         MAXMSGLEN(4194304) CVTMSG(*NO) +
                         HRTBTINTVL(300) NPMSPEED(*FAST)

    /* Defaults for a RECEIVER channel */
            CRTMQMCHL  CHLNAME(SYSTEM.DEF.RECEIVER) CHLTYPE(*RCVR) +
                         REPLACE(*YES) TRPTYPE(*TCP) TEXT(' ') +
                         BATCHSIZE(50) +
                         SCYEXIT(*NONE) SCYUSRDATA(*NONE) +
                         SNDEXIT(*NONE) SNDUSRDATA(*NONE) +
                         RCVEXIT(*NONE) RCVUSRDATA(*NONE) +
                         MSGEXIT(*NONE) MSGUSRDATA(*NONE) +
                         PUTAUT(*DFT) SEQNUMWRAP(999999999) +
                         MAXMSGLEN(4194304) HRTBTINTVL(300) +
                         NPMSPEED(*FAST)

    /* Defaults for an automatically defined RECEIVER channel */
            CRTMQMCHL  CHLNAME(SYSTEM.AUTO.RECEIVER) CHLTYPE(*RCVR) +
                         REPLACE(*YES) TRPTYPE(*TCP) +
                         TEXT('Auto-defined by') +
                             BATCHSIZE(50)          +
                             SCYEXIT(*NONE) SCYUSRDATA(*NONE) +
                             SNDEXIT(*NONE) SNDUSRDATA(*NONE) +
                             RCVEXIT(*NONE) RCVUSRDATA(*NONE) +
                             MSGEXIT(*NONE) MSGUSRDATA(*NONE) +
                             PUTAUT(*DFT) SEQNUMWRAP(999999999) +
                             MAXMSGLEN(4194304) HRTBTINTVL(300) +
                             NPMSPEED(*FAST)

    /* Defaults for a REQUESTER channel */
            CRTMQMCHL  CHLNAME(SYSTEM.DEF.REQUESTER) +
                             CHLTYPE(*RQSTR) REPLACE(*YES) +
                             TRPTYPE(*TCP) TEXT(' ') CONNAME(' ') +
                             MCANAME(*NONE) BATCHSIZE(50) +
                             SCYEXIT(*NONE) SCYUSRDATA(*NONE) +
                             SNDEXIT(*NONE) SNDUSRDATA(*NONE) +
                             RCVEXIT(*NONE) RCVUSRDATA(*NONE) +
                             MSGEXIT(*NONE) MSGUSRDATA(*NONE) +
                             PUTAUT(*DFT) SEQNUMWRAP(999999999) +
                             MAXMSGLEN(4194304) HRTBTINTVL(300) +
                             NPMSPEED(*FAST)

    /* Defaults for a SVRCONN channel   */
            CRTMQMCHL  CHLNAME(SYSTEM.DEF.SVRCONN) CHLTYPE(*SVRCN) +
                             REPLACE(*YES) TRPTYPE(*TCP) TEXT(' ') +
                             SCYEXIT(*NONE) SCYUSRDATA(*NONE) +
                             SNDEXIT(*NONE) SNDUSRDATA(*NONE) +
                             RCVEXIT(*NONE) RCVUSRDATA(*NONE) +
                             MCAUSRID(*PUBLIC) +
                             MAXMSGLEN(4194304) HRTBTINTVL(300)

    /* Defaults for an automatically defined SVRCONN channel   */
            CRTMQMCHL  CHLNAME(SYSTEM.AUTO.SVRCONN) CHLTYPE(*SVRCN) +
                             REPLACE(*YES) TRPTYPE(*TCP) +
                             TEXT('Auto-defined by') +
                             SCYEXIT(*NONE) SCYUSRDATA(*NONE) +
                             SNDEXIT(*NONE) SNDUSRDATA(*NONE) +
                             RCVEXIT(*NONE) RCVUSRDATA(*NONE) +
                             MCAUSRID(*PUBLIC) +
                             MAXMSGLEN(4194304) HRTBTINTVL(300)

/******************************************************************/
/*       DEFAULT CHANNEL QUEUES                                   */
/*                                                                */
/*   Create the queue definitions used by the Channels            */
/*                                                                */
/******************************************************************/
    /* Channel initiation queue   */
            CRTMQMQ     QNAME('SYSTEM.CHANNEL.INITQ')             +
                        QTYPE(*LCL)  REPLACE(*YES)                +
                                                                  +
                        TEXT('MQSeries Channel initiation queue')      +
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        GETENBL(*YES)  /* Get operations allowed  */+
                        DFTMSGPST(*NO) /* Default non-persistent  */+
                        MAXDEPTH(1000) /* Maximum queue depth     */+
                        MAXMSGLEN(2000)/* Maximum message length  */+
                        SHARE(*NO)     /* Not shareable           */+
                        DFTSHARE(*NO)  /* Input open option       */+
                        USAGE(*NORMAL) /* Normal queue usage      */+
                        TRGENBL(*NO)   /* Trigger control off     */

    /* Channel synchronization queue   */
            CRTMQMQ     QNAME('SYSTEM.CHANNEL.SYNCQ')             +
                          QTYPE(*LCL)  REPLACE(*YES)                +
                                                                    +
                       TEXT('MQSeries Channel synchronization queue') +
                          PUTENBL(*YES)  /* Put operations allowed  */+
                          GETENBL(*YES)  /* Get operations allowed  */+
                          DFTMSGPST(*YES)/* Default persistent      */+
                          MAXDEPTH(20000)                           +
                                         /* Maximum queue depth     */+
                          MAXMSGLEN(20000)                          +
                                         /* Maximum message length  */+
                          SHARE(*YES)    /* Shareable               */+
                          DFTSHARE(*YES) /* Input open option       */+
                          USAGE(*NORMAL) /* Normal queue usage      */+
                          TRGENBL(*NO)   /* Trigger control off     */

/*******************************************************************/
/*       DEFAULT INITIATION QUEUES                                 */
/*                                                                 */
/*   Create the initiation queue definition used by the supplied   */
/*   trigger monitor.                                              */
/*                                                                 */
/*******************************************************************/
    /* Default initiation queue   */
            CRTMQMQ     QNAME('SYSTEM.DEFAULT.INITIATION.QUEUE')    +
                        QTYPE(*LCL)  REPLACE(*YES)                  +
                                                                    +
                        TEXT('MQSeries Default initiation queue')   +
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        GETENBL(*YES)  /* Get operations allowed  */+
                        DFTMSGPST(*NO) /* Default non-persistent  */+
                        MAXDEPTH(1000) /* Maximum queue depth     */+
                        MAXMSGLEN(1000)/* Maximum message length  */+
                        SHARE(*YES)    /* Shareable               */+
                        DFTSHARE(*YES) /* Input open option       */+
                        USAGE(*NORMAL) /* Normal queue usage      */+
                        TRGENBL(*NO)   /* Trigger control off     */

    /* Default initiation queue   */
            CRTMQMQ     QNAME('SYSTEM.CICS.INITIATION.QUEUE')       +
                        QTYPE(*LCL)  REPLACE(*YES)                  +
                                                                    +
                       TEXT('MQSeries Default CICS initiation queue') +
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        GETENBL(*YES)  /* Get operations allowed  */+
                        DFTMSGPST(*NO) /* Default non-persistent  */+
                        MAXDEPTH(1000) /* Maximum queue depth     */+
                        MAXMSGLEN(1000)/* Maximum message length  */+
                        SHARE(*YES)    /* Shareable               */+
                        DFTSHARE(*YES) /* Input open option       */+
                        USAGE(*NORMAL) /* Normal queue usage      */+
                        TRGENBL(*NO)   /* Trigger control off     */

/*******************************************************************/
/*       SYSTEM.ADMIN.COMMAND.QUEUE                                */
/*                                                                 */
/*   Create the system-defined queue through which requests        */
/*   are sent to the Queue Manager                                 */
/*                                                                 */
/*******************************************************************/
            CRTMQMQ   QNAME('SYSTEM.ADMIN.COMMAND.QUEUE')           +
                      QTYPE(*LCL)  REPLACE(*YES)                    +
                                                                    +
                      TEXT('MQSeries administration command queue') +
                      DFTMSGPST(*NO)   /* Default non-persistent  */+
                      MAXDEPTH(3000)   /* Maximum queue depth     */+
                      MAXMSGLEN(9000)  /* Maximum message length  */+
                      SHARE(*NO)       /* not Shareable           */+
                      DFTSHARE(*NO)    /* Input Open Option       */+
                      USAGE(*NORMAL)   /* Normal queue usage      */+
                      TRGTYPE(*NONE)   /* No trigger messages     */

/********************************************************************/
/*       SYSTEM.MQSC.REPLY.QUEUE                                    */
/*                                                                  */
/*   Create reply queue for MQSC - can be LOCAL or MODEL            */
/*                                                                  */
/********************************************************************/
            CRTMQMQ   QNAME('SYSTEM.MQSC.REPLY.QUEUE')              +
                      QTYPE(*MDL)  REPLACE(*YES)                    +
                                                                    +
                      TEXT('MQSeries reply queue')                  +
                      DFNTYPE(*TEMPDYN)/* Temporary dynamic       */+
                      DFTMSGPST(*NO)   /* Default non-persistent  */+
                      MAXDEPTH(3000)   /* Maximum queue depth     */+
                      MAXMSGLEN(9000)  /* Maximum message length  */+
                      SHARE(*NO)       /* not Shareable           */+
                      DFTSHARE(*NO)    /* Input Open Option       */+
                      USAGE(*NORMAL)   /* Normal queue usage      */+
                      TRGTYPE(*NONE)   /* No trigger messages     */

/********************************************************************/
/*       SYSTEM.ADMIN.QMGR.EVENT                                    */
/*                                                                  */
/*   Create the system-defined queue through which Queue Manager    */
/*   related Events are reported.                                   */
/*                                                                  */
/********************************************************************/
            CRTMQMQ   QNAME('SYSTEM.ADMIN.QMGR.EVENT')              +
                      QTYPE(*LCL)  REPLACE(*YES)                    +
                                                                    +
                 TEXT('MQSeries Queue Manager related event queue') +
                      DFTMSGPST(*NO)   /* Default non-persistent  */+
                      MAXDEPTH(3000)   /* Maximum queue depth     */+
                      MAXMSGLEN(9000)  /* Maximum message length  */+
                      SHARE(*NO)       /* not Shareable           */+
                      DFTSHARE(*NO)    /* Input Open Option       */+
                      USAGE(*NORMAL)   /* Normal queue usage      */+
                      TRGTYPE(*NONE)   /* No trigger messages     */

/********************************************************************/
/*       SYSTEM.ADMIN.PERFM.EVENT                                   */
/*                                                                  */
/*   Create the system-defined queue through which performance      */
/*   related Events are reported.                                   */
/*                                                                  */
/********************************************************************/
            CRTMQMQ   QNAME('SYSTEM.ADMIN.PERFM.EVENT')              +
                      QTYPE(*LCL)  REPLACE(*YES)                    +
                                                                    +
                   TEXT('MQSeries performance related event queue') +
                      DFTMSGPST(*NO)   /* Default non-persistent  */+
                      MAXDEPTH(3000)   /* Maximum queue depth     */+
                      MAXMSGLEN(9000)  /* Maximum message length  */+
                      SHARE(*NO)       /* not Shareable           */+
                      DFTSHARE(*NO)    /* Input Open Option       */+
                      USAGE(*NORMAL)   /* Normal queue usage      */+
                      TRGTYPE(*NONE)   /* No trigger messages     */
```

```
/*******************************************************************/
/*       SYSTEM.ADMIN.CHANNEL.EVENT                                */
/*                                                                 */
/*   Create the system-defined queue through which channel         */
/*   related Events are reported.                                  */
/*                                                                 */
/*******************************************************************/
            CRTMQMQ   QNAME('SYSTEM.ADMIN.CHANNEL.EVENT')          +
                      QTYPE(*LCL)  REPLACE(*YES)                   +
                                                                   +
                      TEXT('MQSeries channel related event queue')  +
                      DFTMSGPST(*NO)   /* Default non-persistent  */+
                      MAXDEPTH(3000)   /* Maximum queue depth     */+
                      MAXMSGLEN(9000)  /* Maximum message length  */+
                      SHARE(*NO)       /* not Shareable           */+
                      DFTSHARE(*NO)    /* Input Open Option       */+
                      USAGE(*NORMAL)   /* Normal queue usage      */+
                      TRGTYPE(*NONE)   /* No trigger messages     */

/*******************************************************************/
/*       DEFAULT DEAD QUEUE                                        */
/*                                                                 */
/*   Create a default dead letter queue.                           */
/*                                                                 */
/*******************************************************************/
            CRTMQMQ   QNAME('SYSTEM.DEAD.LETTER.QUEUE')            +
                      QTYPE(*LCL)  REPLACE(*YES)                   +
                                                                   +
                      TEXT('MQSeries default dead letter queue')    +
                      DFTMSGPST(*NO)   /* Default non-persistent  */+
                      MAXDEPTH(640000) /* Maximum queue depth     */+
                      MAXMSGLEN(4194304)                           +
                                       /* Maximum message length  */+
                      SHARE(*YES)      /* Shareable               */+
                      DFTSHARE(*YES)   /* Input Open Option       */+
                      USAGE(*NORMAL)   /* Normal queue usage      */+
                      TRGTYPE(*NONE)   /* No trigger messages     */

/*******************************************************************/
/*                                                                 */
/*   Normal return.                                                */
/*                                                                 */
/*******************************************************************/
             QSYS/GOTO  CMDLBL(EXIT)

/*******************************************************************/
/*                                                                 */
/*   Error return                                                  */
/*                                                                 */
/*******************************************************************/
ERROR:      QSYS/SNDPGMMSG MSGID(CPFA09D) MSGF(QCPFMSG) +
                         MSGDTA(&PGMNAME)
EXIT:       ENDPGM

/*******************************************************************/
/*                                                                 */
/* END OF AMQSDEF4                                                 */
/*                                                                 */
/*******************************************************************/
```

AMQSAMP4

```
/********************************************************************/
/*                                                                  */
/* 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                                     */
/*                                                                  */
/* Statement:     Licensed Materials - Property of IBM              */
/*                                                                  */
/*               5763-MQ2                                           */
/*               (C) Copyright IBM Corporation 1993, 1996.          */
/*                                                                  */
/* Status: Version 3 Release 2.0                                    */
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* 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 has no parameters.                                    */
/*                                                                  */
/********************************************************************/
            PGM


/********************************************************************/
/*      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')               +
                        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')               +
                        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')              +
                        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 */+
                        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')               +
                        QTYPE(*LCL)  REPLACE(*YES)                 +
                                                                   +
                        TEXT('General reply queue')                +
                        DFTMSGPST(*YES)  /* Persistent messages OK  */

/*   Queue used by AMQSINQA                                         */
            CRTMQMQ     QNAME('SYSTEM.SAMPLE.INQ')                 +
                        QTYPE(*LCL)  REPLACE(*YES)                 +
                                                                   +
                        TEXT('queue for AMQSINQA')                 +
                        SHARE(*YES)                /* Shareable   */+
                        DFTMSGPST(*YES)/* Persistent messages OK  */+
                                                                   +
                        TRGENBL(*YES)  /* Trigger control on      */+
                        TRGTYPE(*FIRST)/* Trigger on first message*/+
                        PRCNAME('SYSTEM.SAMPLE.INQPROCESS')        +
                        INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*   Queue used by AMQSSETA                                         */
            CRTMQMQ     QNAME('SYSTEM.SAMPLE.SET')                 +
                        QTYPE(*LCL)  REPLACE(*YES)                 +
                                                                   +
                        TEXT('queue for AMQSSETA')                 +
                        SHARE(*YES)                /* Shareable  */ +
                        DFTMSGPST(*YES)/* Persistent messages OK */ +
                                                                   +
                        TRGENBL(*YES)  /* Trigger control on     */ +
                        TRGTYPE(*FIRST)/* Trigger on first message*/+
                        PRCNAME('SYSTEM.SAMPLE.SETPROCESS')        +
                        INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/*   Queue used by AMQSECHA                                         */
            CRTMQMQ     QNAME('SYSTEM.SAMPLE.ECHO')                +
                        QTYPE(*LCL)  REPLACE(*YES)                 +
                                                                   +
                        TEXT('queue for AMQSECHA')                 +
                        SHARE(*YES)                /* Shareable  */ +
                        DFTMSGPST(*YES)/* Persistent messages OK */ +
                                                                    +
                        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') +
                        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')         +
                        REPLACE(*YES)                               +
                                                                    +
                        TEXT('trigger process for AMQSINQA')        +
                        ENVDATA('JOBPTY(3)') /* Submit parameter */ +
         /** Select the triggered program here   **/                +
                        APPID('AMQSINQA')    /* C     */            +
               /*       APPID('AMQ0INQA')    /* COBOL */            +
               /*       APPID('AMQ1INQ4')    /* RPG - OPM */        +
               /*       APPID('AMQ2INQ4')    /* RPG - ILE */

            CRTMQMPRC   PRCNAME('SYSTEM.SAMPLE.SETPROCESS')         +
                        REPLACE(*YES)                               +
                                                                    +
                        TEXT('trigger process for AMQSSETA')        +
                        ENVDATA('JOBPTY(3)') /* Submit parameter */ +
         /** Select the triggered program here   **/                +
                        APPID('AMQSSETA')    /* C     */            +
               /*       APPID('AMQ0SETA')    /* COBOL */            +
               /*       APPID('AMQ1SET4')    /* RPG - OPM */        +
               /*       APPID('AMQ2SET4')    /* RPG - ILE */

            CRTMQMPRC   PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS')        +
                        REPLACE(*YES)                               +
                                                                    +
                        TEXT('trigger process for AMQSECHA')        +
                        ENVDATA('JOBPTY(3)') /* Submit parameter */ +
         /** Select the triggered program here   **/                +
                        APPID('AMQSECHA')    /* C     */            +
               /*       APPID('AMQ0ECHA')    /* COBOL */            +
               /*       APPID('AMQ1ECH4')    /* RPG - OPM */        +
               /*       APPID('AMQ2ECH4')    /* RPG - ILE */

/********************************************************************/
/*   Normal return.                                                 */
/********************************************************************/
            RETURN
            ENDPGM

/********************************************************************/
/* END OF AMQSAMP4                                                  */
/********************************************************************/
```

AMQSADM4

```
/********************************************************************/
/*                                                                  */
/* Program name: AMQSADM4                                           */
/*                                                                  */
/* Description: Sample CL program defining MQM queues               */
/*              needed to run the administrator application.        */
/*                                                                  */
/* Statement:     Licensed Materials - Property of IBM              */
/*                                                                  */
/*                5763-MQ1                                          */
/*                (C) Copyright IBM Corporation 1993, 1996          */
/*                                                                  */
/* Status: Version 3 Release 7.0                                    */
/*                                                                  */
/********************************************************************/
/*                                                                  */
/* Function:                                                        */
/*                                                                  */
/*                                                                  */
/*   AMQSADM4 is a sample CL program to create or reset the         */
/*   queues and set the authority required to run the               */
/*   MQSeries administration application.                           */
/*                                                                  */
/*      -- Under normal circumstances values here should not need   */
/*         to be changed. The values that can be changed are        */
/*                                                                  */
/*         -- MAXDEPTH on SYSTEM.ADMIN.MQMLIST.userid               */
/*                                                                  */
/*         -- the authority levels granted by GRTMQMAUT.            */
/*                                                                  */
/*         Consult the administration guide for more information   */
/*         on tailoring the administration application.             */
/*                                                                  */
/*   This program, or a similar one, should be run                  */
/*   prior to starting the administrator application.               */
/*                                                                  */
/********************************************************************/
/*                                                                  */
/*   Exceptions signaled:  none                                     */
/*   Exceptions monitored: CPF0001                                  */
/*                                                                  */
/*   AMQSADM4 has one parameter, the userid of the administrator    */
/*   which is being set up.                                         */
/*                                                                  */
/********************************************************************/

/********************************************************************/
/*  Define the user identifier parameter                            */
/********************************************************************/
            PGM        PARM(&USERID)

/********************************************************************/
/*  Define local variables                                          */
/********************************************************************/
            DCL        VAR(&USERID) TYPE(*CHAR) LEN(10)
            DCL        VAR(&QNAME) TYPE(*CHAR) LEN(48)
            DCL        VAR(&QPREFIX) TYPE(*CHAR) VALUE(SYSTEM.ADMIN.)
            DCL        VAR(&COPYTEXT)  TYPE(*CHAR) LEN(50)
            DCL        VAR(&COPYRIGHT) TYPE(*CHAR) LEN(50) +
                         VALUE('5763-MQ1 (C) Copyright IBM +
                         Corporation 1993,1996')
/* force a usage of copyright                                       */
```

```
            QSYS/CHGVAR VAR(&COPYTEXT) VALUE(&COPYRIGHT)

/********************************************************************/
/*      Grant administator authority to MQM objects                 */
/*      Monitor for invalid user identifier                         */
/********************************************************************/
            MONMSG     MSGID(CPF0001) EXEC(GOTO CMDLBL(ERROR))
            GRTMQMAUT  OBJ(*ALL) OBJTYPE(*ALL) USER(&USERID) AUT(*ALL)

/********************************************************************/
/*      SYSTEM.ADMIN.MQMLIST.userid                                 */
/*      This queue stores information about the queue managers      */
/*      being managed, and the control options.                     */
/********************************************************************/
            CHGVAR     VAR(&QNAME) VALUE(&QPREFIX||MQMLIST.||&USERI

            CRTMQMQ     QNAME(&QNAME)                              +
                        QTYPE(*LCL)  REPLACE(*YES)                 +
                                                                   +
                        TEXT('Queue managers managed by +
                        administrator')                            +
                                                                   +
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        DFTPTY(5)      /* Default priority        */+
                        DFTMSGPST(*YES)/* Default msg persistent  */+
                        GETENBL(*YES)  /* Get operations allowed  */+
                        /*****************************************/+
                        /* The max depth governs the number of    */+
                        /* queue managers that can be managed.    */+
                        /*****************************************/+
                        MAXDEPTH(5000) /* Maximum queue depth     */+
                        MAXMSGLEN(256) /* Maximum message length  */+
                        SHARE(*YES)    /* Shareable               */+
                        DFTSHARE(*YES) /* Input open option       */+
                        MSGDLYSEQ(*FIFO) /* Delivery FIFO         */+
                        HDNBKTCNT(*NO) /* Backout not hardened    */+
                        USAGE(*NORMAL) /* Normal queue usage      */+
                                                                   +
                        TRGENBL(*NO)   /* Trigger control off     */+
                        TRGTYPE(*NONE) /* No trigger messages     */+
                        TRGDEPTH(1)    /* Trigger on first message*/+
                        TRGMSGPTY(0)   /* Trigger on any priority */+
                        TRGDATA(' ')   /* Trigger data            */+
                        PRCNAME(' ')   /* Process name if trigger */+
                        INITQNAME(' ') /* Initiation queue name   */+
                                                                   +
                     RTNITV(999999999) /* Retention time (hours)  */+
                        BKTTHLD(5)     /* Backout threshold       */+
                        BKTQNAME(' ')  /* Backout requeue name    */


/********************************************************************/
/*      SYSTEM.ADMIN.REPLYQ.userid                                  */
/*      This queue is the reply to queue for requests sent to a     */
/*      command server queue and the administrator datastore.       */
/********************************************************************/
            CHGVAR     VAR(&QNAME) VALUE(&QPREFIX||REPLYQ.||&USERID

            CRTMQMQ     QNAME(&QNAME)                              +
                        QTYPE(*LCL)  REPLACE(*YES)                 +
                                                                   +
                        TEXT('System admin application reply to +
                        Queue')   +
                                                                   +
                                    PUTENBL(*YES)  /* Put operations allowed  */+
                                    DFTPTY(5)      /* Default priority        */+
                                    DFTMSGPST(*YES)/* Default msg-persistent  */+
                                    GETENBL(*YES)  /* Get operations allowed  */+
                                    MAXDEPTH(5000) /* Maximum queue depth     */+
                                    MAXMSGLEN(65000)/* Maximum message length */+
                                    SHARE(*YES)    /* Shareable               */+
                                    DFTSHARE(*YES) /* Input open option       */+
                                    MSGDLYSEQ(*FIFO) /* delivery fifo         */+
                                    HDNBKTCNT(*NO) /* Backout not hardened    */+
                                    USAGE(*NORMAL) /* Normal queue usage      */+
                                                                               +
                                    TRGENBL(*NO)   /* Trigger control off     */+
                                    TRGTYPE(*NONE) /* No trigger messages     */+
                                    TRGDEPTH(1)    /* Trigger on first message*/+
                                    TRGMSGPTY(0)   /* Trigger on any priority */+
                                    TRGDATA(' ')   /* Trigger data            */+
                                    PRCNAME(' ')   /* Process name if trigger */+
                                    INITQNAME(' ') /* Initiation queue name   */+
                                                                               +
                                 RTNITV(999999999) /* Retention time (hours)  */+
                                    BKTTHLD(5)     /* Backout threshold       */+
                                    BKTQNAME(' ')  /* Backout requeue name    */


/*********************************************************************/
/*       SYSTEM.ADMIN.RESPQ.userid                                  */
/*       This queue is used to stored copies of the remote commands */
/*       sent to a command server queue and the responses received  */
/*       in reply.                                                  */
/*********************************************************************/
            CHGVAR     VAR(&QNAME) VALUE(&QPREFIX||RESPQ.||&USERID)

            CRTMQMQ     QNAME(&QNAME)                                 +
                        QTYPE(*LCL)  REPLACE(*YES)                    +
                                                                      +
                        TEXT('System admin application response +
                        queue')  +
                                                                      +
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        DFTPTY(5)      /* Default priority        */+
                        DFTMSGPST(*YES)/* Default msg-persistent  */+
                        GETENBL(*YES)  /* Get operations allowed  */+
                        MAXDEPTH(5000) /* Maximum queue depth     */+
                        MAXMSGLEN(65000)/* Maximum message length */+
                        SHARE(*YES)    /* Shareable               */+
                        DFTSHARE(*YES) /* Input open option       */+
                        MSGDLYSEQ(*FIFO) /* delivery fifo         */+
                        HDNBKTCNT(*NO) /* Backout not hardened    */+
                        USAGE(*NORMAL) /* Normal queue usage      */+
                                                                   +
                        TRGENBL(*NO)   /* Trigger control off     */+
                        TRGTYPE(*NONE) /* No trigger messages     */+
                        TRGDEPTH(1)    /* Trigger on first message*/+
                        TRGMSGPTY(0)   /* Trigger on any priority */+
                        TRGDATA(' ')   /* Trigger data            */+
                        PRCNAME(' ')   /* Process name if trigger */+
                        INITQNAME(' ') /* Initiation queue name   */+
                                                                   +
                     RTNITV(999999999) /* Retention time (hours)  */+
                        BKTTHLD(5)     /* Backout threshold       */+
                        BKTQNAME(' ')  /* Backout requeue name    */

/*********************************************************************/
```

```
/*       SYSTEM.ADMIN.EXCEPTION.QUEUE                                */
/*       This queue is used for diagnostic purposes. Invalid         */
/*       messages and other diagnostic information is written to     */
/*       this queue.                                                 */
/*******************************************************************/
            CRTMQMQ     QNAME('SYSTEM.ADMIN.EXCEPTION.QUEUE')       +
                        QTYPE(*LCL)  REPLACE(*YES)                  +
                                                                    +
                        TEXT('System admin application exception +
                        queue') +
                                                                    +
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        DFTPTY(5)      /* Default priority        */+
                        DFTMSGPST(*YES)/* Default msg-persistent  */+
                        GETENBL(*YES)  /* Get operations allowed  */+
                        MAXDEPTH(5000) /* Maximum queue depth     */+
                        MAXMSGLEN(65000)/* Maximum message length */+
                        SHARE(*YES)    /* Shareable               */+
                        DFTSHARE(*YES) /* Input open option       */+
                        MSGDLYSEQ(*FIFO) /* delivery sequence     */+
                        HDNBKTCNT(*NO) /* Backout not hardened    */+
                        USAGE(*NORMAL) /* Normal queue usage      */+
                                                                    +
                        TRGENBL(*NO)   /* Trigger control off     */+
                        TRGTYPE(*NONE) /* No trigger messages     */+
                        TRGDEPTH(1)    /* Trigger on first message*/+
                        TRGMSGPTY(0)   /* Trigger on any priority */+
                        TRGDATA(' ')   /* Trigger data            */+
                        PRCNAME(' ')   /* Process name if trigger */+
                        INITQNAME(' ') /* Initiation queue name   */+
                                                                    +
                     RTNITV(999999999) /* Retention time (hours)  */+
                        BKTTHLD(5)     /* Backout threshold       */+
                        BKTQNAME(' ')  /* Backout requeue name    */


/*******************************************************************/
/*       SYSTEM.ADMIN.AMQMDATA.QUEUE                                 */
/*       This queue is the data store server queue. The data store  */
/*       keeps information on objects on queue managers being       */
/*       administrated by administrators working on this machine.   */
/*******************************************************************/
            CRTMQMQ     QNAME('SYSTEM.ADMIN.AMQMDATA.QUEUE')       +
                        QTYPE(*LCL)  REPLACE(*YES)                  +
                                                                    +
                        TEXT('System Admin Application Data store +
                        Queue') +
                        PUTENBL(*YES)  /* Put operations allowed  */+
                        DFTPTY(5)      /* Default priority        */+
                        DFTMSGPST(*YES)/* Default msg-persistent  */+
                        GETENBL(*YES)  /* Get operations allowed  */+
                        MAXDEPTH(5000) /* Maximum queue depth     */+
                        MAXMSGLEN(65000)/* Maximum message length */+
                        SHARE(*YES)    /* Shareable               */+
                        DFTSHARE(*YES) /* Input open option       */+
                        MSGDLYSEQ(*PTY) /* Delivery sequence prty */+
                        HDNBKTCNT(*NO) /* Backout not hardened    */+
                        USAGE(*NORMAL) /* Normal queue usage      */+
                                                                    +
                        TRGENBL(*NO)   /* Trigger control off     */+
                        TRGTYPE(*NONE) /* No trigger messages     */+
                        TRGDEPTH(1)    /* Trigger on first message*/+
                        TRGMSGPTY(0)   /* Trigger on any priority */+
                        TRGDATA(' ')   /* Trigger data            */+
                        PRCNAME(' ')   /* Process name if trigger */+
                        INITQNAME(' ') /* Initiation queue name   */+
                                                                   +
                     RTNITV(999999999) /* Retention time (hours)  */+
                        BKTTHLD(5)     /* Backout threshold       */+
                        BKTQNAME(' ')  /* Backout requeue name    */

/*********************************************************************/
/*       Grant administator authority to MQM objects                 */
/*       just created.                                               */
/*********************************************************************/
            GRTMQMAUT  OBJ(*ALL) OBJTYPE(*ALL) USER(&USERID) AUT(*ALL)

/*********************************************************************/
/*                                                                   */
/*   Normal return.                                                  */
/*                                                                   */
/*********************************************************************/
            RETURN

/*********************************************************************/
/*                                                                   */
/*   ERROR RETURN,                                                   */
/*                                                                   */
/*********************************************************************/
ERROR:
            SNDPGMMSG  MSG('Errors occurred, check the joblog.')
            RETURN

/*********************************************************************/
/*  END OF PROGRAM                                                   */
/*********************************************************************/
            ENDPGM

/*********************************************************************/
/*                                                                   */
/* END OF AMQSADM4                                                   */
/*                                                                   */
/*********************************************************************/
```

AMQSCOMA

```
*******************************************************************/
*                                                                 */
* Program name: AMQSCOMA                                          */
*                                                                 */
* Description: Sample MQSC source defining MQM queues             */
*              Can be processed, with changes as needed, after    */
*              starting the MQM                                   */
*                                                                 */
* Statement:     Licensed Materials - Property of IBM             */
*                                                                 */
*                33H2205, 5622-908                                */
*                33H2267, 5765-623                                */
*                29H0990, 5697-176                                */
*                (C) Copyright IBM Corp. 1994, 1998               */
*                                                                 */
*******************************************************************/
* Function:                                                       */
*                                                                 */
*                                                                 */
*   AMQSCOMA is a sample MQSC file to create or reset the         */
*   MQI resources of the Message Queue Manager (MQM).             */
*   It includes the following.                                    */
*                                                                  */
*      -- specification of the standard default attributes         */
*                                                                  */
*      -- definition of standard MQM queues                        */
*                                                                  */
*   This file, or a similar one, can be processed when the MQM     */
*   is started - it creates the objects if missing, or resets      */
*   their attributes to the prescribed values.                     */
*                                                                  */
*******************************************************************/
*******************************************************************/
*                                                                  */
*   AMQSCOMA is sample data for the MQSC utility                   */
*                                                                  */
*******************************************************************/


*******************************************************************/
*       DEFAULT ALIAS QUEUE                                        */
*                                                                  */
*   Define the default alias queue.                                */
*     REPLACE allows this command to reset the attributes          */
*     to known values.                                             */
*                                                                  */
*     Note: because it's a default queue, all attributes           */
*     applicable to an alias queue are explicit, and all           */
*     users have READ access.                                      */
*                                                                  */
*******************************************************************/
    DEFINE QALIAS('SYSTEM.DEFAULT.ALIAS.QUEUE') REPLACE +

*   Queue description
           DESCR(' ')  +
*   Put operations allowed
           PUT(ENABLED)     +
*   Default priority
           DEFPRTY(0)       +
*   Default non-persistent
           DEFPSIST(NO)     +
*   Get operations allowed
           GET(ENABLED)     +
*   Base queue name
           TARGQ(' ')       +
*   Scope
           SCOPE(QMGR)

*******************************************************************/
*       DEFAULT LOCAL QUEUE                                        */
*                                                                  */
*   Define the default local queue.                                */
*     REPLACE allows this command to reset the attributes          */
*     to known values.                                             */
*                                                                  */
*     Note: because it's a default queue, all attributes           */
*     applicable to a local queue are explicit, and all            */
*     users have READ access.                                      */
*                                                                  */
*******************************************************************/
    DEFINE QLOCAL('SYSTEM.DEFAULT.LOCAL.QUEUE') REPLACE  +

*   Queue description
           DESCR(' ')         +
*   Put operations allowed
```

```
          PUT(ENABLED)       +
*   Default priority
          DEFPRTY(0)         +
*   Default non-persistent
          DEFPSIST(NO)       +
*   Get operations allowed
          GET(ENABLED)       +
*   If a transmission queue is serviced by a channel to a queue manager
*   that supports distribution lists then the channel will dynamically set
*   this bit to supported. Queues should normally be defined as DISABLED.
          DISTL(NO) +
*   Maximum queue depth
          MAXDEPTH(5000)     +
*   Maximum message length
          MAXMSGL(4194304)   +
*   Shareable
          SHARE              +
*   Input open option
          DEFSOPT(SHARED)    +
*   Delivery by priority
          MSGDLVSQ(PRIORITY) +
*   Backout not hardened
          NOHARDENBO         +
*   Normal queue usage
          USAGE(NORMAL)      +

*   Trigger control off
          NOTRIGGER          +

*** Next six defaults apply if trigger is made active
*   otherwise trigger on first message
          TRIGTYPE(FIRST)    +
*   Trigger on first message - default for queues with TRIGTYPE(DEPTH)
          TRIGDPTH(1)        +
*   Trigger on any priority
          TRIGMPRI(0)        +
*   Trigger data
          TRIGDATA(' ')      +
*   Process name if trigger
          PROCESS(' ')       +
*   Initiation queue name
          INITQ(' ')         +

*** Other attributes can be tested, but MQM takes no action
*   Retention time (hours)
          RETINTVL(999999999) +
*   Backout threshold
          BOTHRESH(0)        +
*   Backout requeue name
          BOQNAME(' ')       +
*   Scope
          SCOPE(QMGR)        +
*   Queue depth high event threshold
          QDEPTHHI(80)       +
*   Queue depth low event threshold
          QDEPTHLO(20)       +
*   Queue full event control
          QDPMAXEV(ENABLED)  +
*   Queue depth high event control
          QDPHIEV(DISABLED)  +
*   Queue depth low event control
          QDPLOEV(DISABLED)  +
*   Queue service interval event threshold
```

```
           QSVCINT(999999999) +
*   Queue service interval event control
           QSVCIEV(NONE)

******************************************************************/
*       DEFAULT MODEL QUEUE                                      */
*                                                                */
*   Define the default model queue.                              */
*     REPLACE allows this command to reset the attributes        */
*     to known values.                                           */
*                                                                */
*     Note: because it's a default queue, all attributes         */
*     applicable to a model queue are explicit, and all          */
*     users have READ access.                                    */
*                                                                */
******************************************************************/
    DEFINE QMODEL('SYSTEM.DEFAULT.MODEL.QUEUE') REPLACE  +

*   Queue description
           DESCR(' ')         +
*   Put operations allowed
           PUT(ENABLED)       +
*   Default priority
           DEFPRTY(0)         +
*   Default non-persistent
           DEFPSIST(NO)       +
*   If a transmission queue is serviced by a channel to a queue manager
*   that supports distribution lists then the channel will dynamically set
*   this bit to supported. Queues should normally be defined as DISABLED.
           DISTL(NO) +
*   Get operations allowed
           GET(ENABLED)       +

*   Maximum queue depth
           MAXDEPTH(5000)     +
*   Maximum message length
           MAXMSGL(4194304)   +
*   Not shareable
           SHARE            +
*   Input open option
           DEFSOPT(SHARED)    +
*   Delivery by priority
           MSGDLVSQ(PRIORITY) +
*   Backout not hardened
           NOHARDENBO         +
*   Normal queue usage
           USAGE(NORMAL)      +
*   Temporary dynamic queue created when model is opened
           DEFTYPE(TEMPDYN)   +

*   Trigger control off
           NOTRIGGER          +

*** Next six defaults apply if trigger is made active
*   Trigger on first message
           TRIGTYPE(FIRST)    +
*   Trigger on first message for TRIGTYPE(DEPTH) too
           TRIGDPTH(1)        +
*   Trigger on any priority
           TRIGMPRI(0)        +
*   Trigger data
           TRIGDATA(' ')      +
*   Process name if trigger
```

```
           PROCESS(' ')       +
*   Initiation queue name
           INITQ(' ')         +

*** Other attributes can be tested, but MQM takes no action
*   Retention time (hours)
           RETINTVL(999999999) +
*   Backout threshold
           BOTHRESH(0)        +
*   Backout requeue name
           BOQNAME(' ')       +
*   Queue depth high event threshold
           QDEPTHHI(80)       +
*   Queue depth low event threshold
           QDEPTHLO(20)       +
*   Queue full event control
           QDPMAXEV(ENABLED)  +
*   Queue depth high event control
           QDPHIEV(DISABLED)  +
*   Queue depth low event control
           QDPLOEV(DISABLED)  +
*   Queue service interval event threshold
           QSVCINT(999999999) +
*   Queue service interval event control
           QSVCIEV(NONE)

**********************************************************************/
*       DEFAULT REMOTE QUEUE                                         */
*                                                                    */
*   Create the default remote queue.                                 */
*     REPLACE allows this command to reset the attributes            */
*     to known values.                                               */
*                                                                    */
*     Note: because it's a default queue, all attributes             */
*     applicable to a remote queue are explicit, and all             */
*     users have READ access.                                        */
*                                                                    */
**********************************************************************/
    DEFINE QREMOTE('SYSTEM.DEFAULT.REMOTE.QUEUE') REPLACE +

*   Queue description
           DESCR(' ')         +
*   Put operations allowed
           PUT(ENABLED)       +
*   Default priority
           DEFPRTY(0)         +
*   Default non-persistent
           DEFPSIST(NO)       +
*   No transmission queue
           XMITQ(' ')         +
*   Queue name
           RNAME(' ')         +
*   Name of remote queue manager
           RQMNAME(' ')       +
*   Scope
           SCOPE(QMGR)

**********************************************************************/
*       DEFAULT MQI PROCESS OBJECT                                   */
*                                                                    */
*   Create the default MQI process object                            */
*     REPLACE allows this command to reset the attributes            */
*     to known values.                                               */
*                                                                    */
*     Note: Because it's a default object, all attributes          */
*     applicable to a process object are explicit, and all         */
*     users have READ access.                                      */
*                                                                  */
*******************************************************************/
    DEFINE PROCESS('SYSTEM.DEFAULT.PROCESS')   REPLACE +

*   Object description
            DESCR(' ')        +
*   Application type (non-specific)
            APPLTYPE(DEF)      +
*   Application identifier
            APPLICID(' ')      +
*   User data
            USERDATA(' ')      +
*   Environment data
            ENVRDATA(' ')

*******************************************************************/
*       DEFAULT CHANNEL DEFINITIONS                                */
*                                                                  */
*   Create the defaults for each of the channel types              */
*     REPLACE allows this command to reset the attributes          */
*     to known values.                                             */
*                                                                  */
*******************************************************************/

*** Defaults for a sender channel
    DEFINE CHANNEL(SYSTEM.DEF.SENDER) CHLTYPE(SDR) +
            TRPTYPE(TCP) REPLACE DESCR(' ')     +
            BATCHSZ(50) DISCINT(6000)  +
            SHORTTMR(60) SHORTRTY(10) +
            LONGTMR(1200) LONGRTY(999999999) +
            SEQWRAP(999999999) MAXMSGL(4194304) +
            CONVERT(NO) XMITQ(' ') CONNAME(' ') +
            HBINT(300) NPMSPEED(FAST)

*** Defaults for a server channel
    DEFINE CHANNEL(SYSTEM.DEF.SERVER) CHLTYPE(SVR) +
            TRPTYPE(TCP) REPLACE DESCR(' ')     +
            BATCHSZ(50) DISCINT(6000)  +
            SHORTTMR(60) SHORTRTY(10) +
            LONGTMR(1200) LONGRTY(999999999) +
            SEQWRAP(999999999) MAXMSGL(4194304) +
            CONVERT(NO) XMITQ(' ') +
            HBINT(300) NPMSPEED(FAST)

*** Defaults for a receiver channel
    DEFINE CHANNEL(SYSTEM.DEF.RECEIVER) CHLTYPE(RCVR) +
            TRPTYPE(TCP) REPLACE DESCR(' ')     +
            BATCHSZ(50) +
            PUTAUT(DEF) SEQWRAP(999999999) +
            MAXMSGL(4194304) +
            HBINT(300) NPMSPEED(FAST)

*** Defaults for an automatically defined receiver channel
    DEFINE CHANNEL(SYSTEM.AUTO.RECEIVER) CHLTYPE(RCVR) +
            TRPTYPE(TCP) REPLACE DESCR('Auto-defined by') +
            BATCHSZ(50) +
            PUTAUT(DEF) SEQWRAP(999999999) +
            MAXMSGL(4194304) +
            HBINT(300) NPMSPEED(FAST)

*** Defaults for a requester channel
```

```
    DEFINE CHANNEL(SYSTEM.DEF.REQUESTER) +
            CHLTYPE(RQSTR) TRPTYPE(TCP) +
            REPLACE DESCR(' ') +
            BATCHSZ(50) PUTAUT(DEF) +
            SEQWRAP(999999999) MAXMSGL(4194304) +
            CONNAME(' ') +
            HBINT(300) NPMSPEED(FAST)

*** Defaults for a svrconn channel
    DEFINE CHANNEL(SYSTEM.DEF.SVRCONN) CHLTYPE(SVRCONN) +
            TRPTYPE(TCP) REPLACE DESCR(' ') +
            MCAUSER('*PUBLIC') +
            MAXMSGL(4194304) +
            HBINT(300)

*** Defaults for an automatically defined svrconn channel
    DEFINE CHANNEL(SYSTEM.AUTO.SVRCONN) CHLTYPE(SVRCONN) +
            TRPTYPE(TCP) REPLACE DESCR('Auto-defined by') +
            MCAUSER('*PUBLIC') +
            MAXMSGL(4194304) +
            HBINT(300)

*********************************************************************/
*       DEFAULT CHANNEL QUEUES                                      */
*                                                                   */
*   Create the queue definitions used by the Channels               */
*                                                                   */
*********************************************************************/
    DEFINE QLOCAL('SYSTEM.CHANNEL.INITQ') REPLACE +
            DESCR('MQSeries Channel initiation queue')    +

*   Put operations allowed
            PUT(ENABLED)      +
*   Get operations allowed
            GET(ENABLED)      +
*   Default non-persistent
            DEFPSIST(NO)      +
*   Maximum queue depth
            MAXDEPTH(1000)    +
*   Maximum message length
            MAXMSGL(2000)     +
*   Not shareable
            NOSHARE           +
*   Input open option
            DEFSOPT(EXCL)     +
*   Normal queue usage
            USAGE(NORMAL)     +
*   No trigger messages
            TRIGTYPE(NONE)

    DEFINE QLOCAL('SYSTEM.CHANNEL.SYNCQ') REPLACE +
            DESCR('MQSeries Channel synchronization queue')    +

*   Put operations allowed
            PUT(ENABLED)      +
*   Get operations allowed
            GET(ENABLED)      +
*   Default non-persistent
            DEFPSIST(YES)     +
*   Maximum queue depth
            MAXDEPTH(20000)   +
*   Maximum message length
            MAXMSGL(20000)    +
*   Shareable
```

```
                SHARE            +
*   Input open option
                DEFSOPT(SHARED)   +
*   Normal queue usage
                USAGE(NORMAL)     +
*   No trigger messages
                TRIGTYPE(NONE)


*******************************************************************/
*       DEFAULT INITIATION QUEUES                                 */
*                                                                 */
*   Create the initiation queue definitions used as defaults      */
*   by the supplied trigger monitors                              */
*                                                                 */
*******************************************************************/
    DEFINE QLOCAL('SYSTEM.DEFAULT.INITIATION.QUEUE') REPLACE +
            DESCR('MQSeries Default initiation queue')    +

*   Put operations allowed
                PUT(ENABLED)      +
*   Get operations allowed
                GET(ENABLED)      +
*   Default non-persistent
                DEFPSIST(NO)      +
*   Maximum queue depth
                MAXDEPTH(1000)    +
*   Maximum length = size of trigger message
                MAXMSGL(1000)     +
*   Shareable
                SHARE             +
*   Input open option
                DEFSOPT(SHARED)   +
*   Normal queue usage
                USAGE(NORMAL)     +
*   No trigger messages
                TRIGTYPE(NONE)

    DEFINE QLOCAL('SYSTEM.CICS.INITIATION.QUEUE') REPLACE +
            DESCR('MQSeries Default CICS initiation queue')    +

*   Put operations allowed
                PUT(ENABLED)      +
*   Get operations allowed
                GET(ENABLED)      +
*   Default non-persistent
                DEFPSIST(NO)      +
*   Maximum queue depth
                MAXDEPTH(1000)    +
*   Maximum length = size of trigger message
                MAXMSGL(1000)     +
*   Shareable
                SHARE             +
*   Input open option
                DEFSOPT(SHARED)   +
*   Normal queue usage
                USAGE(NORMAL)     +
*   No trigger messages
                TRIGTYPE(NONE)

*******************************************************************/
*       SYSTEM.ADMIN.COMMAND.QUEUE                                */
*                                                                 */
*   Create the system-defined queue through which requests        */
*   are sent to the Queue Manager                                   */
*                                                                   */
*********************************************************************/
    DEFINE QLOCAL('SYSTEM.ADMIN.COMMAND.QUEUE') REPLACE +
            DESCR('MQSeries administration command queue')    +

*   Default non-persistent
            DEFPSIST(NO)      +
*   Maximum queue depth
            MAXDEPTH(3000)    +
*   Maximum message length
            MAXMSGL(9000)     +
*   Not shareable
            NOSHARE           +
*   Input open option
            DEFSOPT(EXCL)     +
*   Normal queue usage
            USAGE(NORMAL)     +
*   No trigger messages
            TRIGTYPE(NONE)

*********************************************************************/
*       SYSTEM.MQSC.REPLY.QUEUE                                     */
*                                                                   */
*   Create reply queue for MQSC - can be LOCAL or MODEL             */
*                                                                   */
*********************************************************************/
    DEFINE QMODEL('SYSTEM.MQSC.REPLY.QUEUE') REPLACE +
            DESCR('MQSC reply queue')    +

*   Temporary dynamic queue created from this model
            DEFTYPE(TEMPDYN)  +
*   Default non-persistent
            DEFPSIST(NO)      +
*   Maximum queue depth
            MAXDEPTH(3000)    +
*   Maximum message length
            MAXMSGL(9000)     +
*   Not shareable
            NOSHARE           +
*   Input open option
            DEFSOPT(EXCL)     +
*   Normal queue usage
            USAGE(NORMAL)     +
*   No trigger messages
            TRIGTYPE(NONE)


*********************************************************************/
*       SYSTEM.ADMIN.QMGR.EVENT                                     */
*                                                                   */
*   Create the system-defined queue through which Queue Manager    */
*   related Events are reported.                                    */
*                                                                   */
*********************************************************************/
    DEFINE QLOCAL('SYSTEM.ADMIN.QMGR.EVENT') REPLACE +
            DESCR('MQSeries Queue Manager related Event Queue')    +

*   Default non-persistent
            DEFPSIST(NO)      +
*   Maximum queue depth
            MAXDEPTH(3000)    +
*   Maximum message length
```

```
            MAXMSGL(9000)     +
*   Not shareable
            NOSHARE           +
*   Input open option
            DEFSOPT(EXCL)     +
*   Normal queue usage
            USAGE(NORMAL)     +
*   No trigger messages
            TRIGTYPE(NONE)

********************************************************************/
*       SYSTEM.ADMIN.PERFM.EVENT                                   */
*                                                                  */
*   Create the system-defined queue through which performance      */
*   related events are reported.                                   */
*                                                                  */
********************************************************************/
    DEFINE QLOCAL('SYSTEM.ADMIN.PERFM.EVENT') REPLACE +
            DESCR('MQSeries performance related event Queue')    +

*   Default non-persistent
            DEFPSIST(NO)      +
*   Maximum queue depth
            MAXDEPTH(3000)    +
*   Maximum message length
            MAXMSGL(9000)     +
*   Not shareable
            NOSHARE           +
*   Input open option
            DEFSOPT(EXCL)     +
*   Normal queue usage
            USAGE(NORMAL)     +
*   No trigger messages
            TRIGTYPE(NONE)

********************************************************************/
*       SYSTEM.ADMIN.CHANNEL.EVENT                                 */
*                                                                  */
*   Create the system-defined queue through which Channel          */
*   related Events are reported.                                   */
*                                                                  */
********************************************************************/
    DEFINE QLOCAL('SYSTEM.ADMIN.CHANNEL.EVENT') REPLACE +
            DESCR('MQSeries Channel related Event Queue')    +

*   Default non-persistent
            DEFPSIST(NO)      +
*   Maximum queue depth
            MAXDEPTH(3000)    +
*   Maximum message length
            MAXMSGL(9000)     +
*   Not shareable
            NOSHARE           +
*   Input open option
            DEFSOPT(EXCL)     +
*   Normal queue usage
            USAGE(NORMAL)     +
*   No trigger messages
            TRIGTYPE(NONE)

********************************************************************/
*       DEFAULT DEAD QUEUE                                         */
*                                                                  */
*   Create a default local dead letter queue.                      */
*                                                                    */
*                                                                    */
**********************************************************************/
    DEFINE QLOCAL('SYSTEM.DEAD.LETTER.QUEUE') REPLACE  +

*   Queue description
           DESCR('MQSeries default dead letter queue') +
*   Default non-persistent
           DEFPSIST(NO)       +
*   Maximum queue depth
           MAXDEPTH(640000)   +
*   Maximum message length
           MAXMSGL(4194304)   +
*   Shareable
           SHARE              +
*   Input open option
           DEFSOPT(SHARED)    +
*   Normal queue usage
           USAGE(NORMAL)      +
*   No trigger messages
           TRIGTYPE(NONE)

**********************************************************************/
*                                                                    */
* END OF AMQSCOMA                                                    */
*                                                                    */
**********************************************************************/
```

Appendix B. Administration utility – example tasks

This appendix contains various example tasks.

Creating a local queue

To create a local queue, you:

1. Type STRMQMADM on the command line and press the `Enter` key.

   Note: If this is the first time that you have used the Administration Utility, you must add the names of the queue managers you are going to work with to your list of queue managers on the main menu. See “Working with the Administration utility” on page 44 for further information.

2. Type `8` in the option field next to the queue manager that you want to work with, and press the `Enter` key.

3. Type a forward slash character in the `Object types` field to select the MQSeries for AS/400 object type, or types, that you want to work with.

   All queue manager objects of the types you select will be displayed in the Work with Objects list (see Figure 24 on page 270).

4. To create a new local queue, you use the blank fields at the top of the list:

   a. Type `1` in the `Opt` field.
   b. Type `*que` in the `Type` field.
   c. Type `*lcl` in the `Sub` field.
   d. Type the name of the new queue in the `Object name` field.

   Note: You can use lower case characters, but you *must* enclose the name in apostrophes.

5. Press the `Enter` key. The Create MQM Queue panel is displayed with the settings all set to `*SYSDFTQ`. When `*SYSDFTQ` is specified, the value of the attribute is taken from the system default object for the relevant object type, in this case, the `SYSTEM.DEFAULT.LOCAL.QUEUE`.

Notes:

1. Details of the default settings for all the objects supplied with MQSeries for AS/400 are shown in “AMQSDEF4” on page 241. To change the default settings you must have loaded the samples tape. Instructions for doing this are given in “Standard AS/400 installation procedure” on page 15.

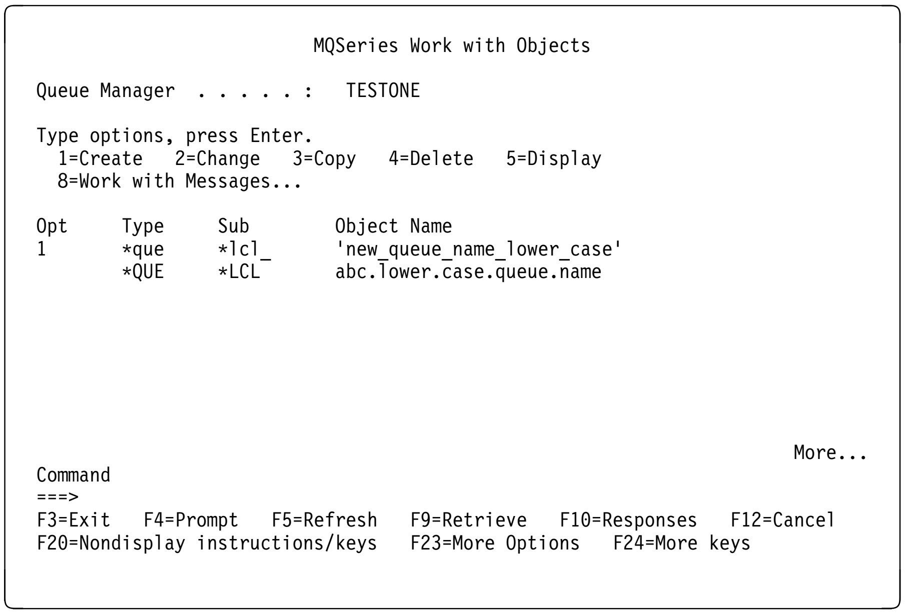
2. To create a remote queue you type `*RMT` in the `Sub` field.

Setting up a link between two queue managers

Notes:

1. The names of the two queue managers used in this example are `TEST1` and `TEST4`.

2. For the default transmission queue, the transmission queue name (*TMQNAME*) parameter, *must* be the same name as the queue manager with which it is associated.



```
                             MQSeries Work with Objects

Queue Manager  . . . . . :   TESTONE

Type options, press Enter.
  1=Create   2=Change   3=Copy   4=Delete   5=Display
  8=Work with Messages...

Opt     Type     Sub        Object Name
1       *que     *lcl_      'new_queue_name_lower_case'
        *QUE     *LCL       abc.lower.case.queue.name







                                                                       More...
Command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Responses   F12=Cancel
F20=Nondisplay instructions/keys   F23=More Options   F24=More keys
```

Figure 24. MQSeries Work with Objects panel

3. A convention of `a_TO_b` is used for channel names where `a` and `b` are the queue-manager names or an abbreviation of the queue-manager name.

Creating the required objects

1. Setup the required objects on `TEST1`:

   a. Create the receiver channel by using the following command:

   ```
   CRTMQMCHL  CHLNAME(TEST4_TO_TEST1) CHLTYPE(*RCVR) +
                TRPTYPE(*TCP)
   ```

   b. Create the sender channel by using the following command:

   ```
   CRTMQMCHL  CHLNAME(TEST1_TO_TEST4) CHLTYPE(*SDR) +
                TRPTYPE(*TCP) CONNAME(TEST4) TMQNAME(TEST4) +
                DSCITV(36000) MAXMSGLEN(65000)
   ```

   c. Create the transmission queue by using the following command:

   ```
   CRTMQMQ    QNAME(TEST4) QTYPE(*LCL) REPLACE(*YES) +
                MAXMSGLEN(65000) USAGE(*TMQ)
   ```

2. Setup the required objects on `TEST4`:

   a. Create the receiver channel by using the following command:

   ```
   CRTMQMCHL  CHLNAME(TEST1_TO_TEST4) CHLTYPE(*RCVR) +
                TRPTYPE(*TCP)
   ```

   b. Create the sender channel by using the following command:

   ```
   CRTMQMCHL  CHLNAME(TEST4_TO_TEST1) CHLTYPE(*SDR) +
                TRPTYPE(*TCP) CONNAME(TEST1) TMQNAME(TEST1) +
                DSCITV(36000) MAXMSGLEN(65000)
   ```

   c. Create the transmission queue by using the following command:

   ```
   CRTMQMQ    QNAME(TEST1) QTYPE(*LCL) REPLACE(*YES) +
                MAXMSGLEN(65000) USAGE(*TMQ)
   ```

Verifying the setup

Table 7 (Page 1 of 2). Queue manager link verification

On System TEST1	On System TEST4
Start a listener by issuing the STRMQMLSR command. This assumes that you are using port 1414. Check the joblog to see if the listener started correctly.	Start a listener by issuing the STRMQMLSR command. This assumes that you are using port 1414. Check the joblog to see if the listener started correctly.
Work with the channels by issuing the WRKMQMCHL command with the option TEST*.	Work with the channels by issuing the WRKMQMCHL command with the option TEST*.
On the MQM Work with channels panel, enter option 13 (Ping) next to TEST1_TO_4. TCP Sender channel. You should get PING MQM channel complete. If there is a problem, an error message is issued indicating the area of the problem. Correct the problem and try again.	On the MQM Work with channels panel, enter option 13 (Ping) next to TEST4_TO_1. TCP Sender channel. You should get PING MQM channel complete. If there is a problem, an error message is issued indicating the area of the problem. Correct the problem and try again.
Enter option 14 (Start) next to TEST1_TO_4. TCP Sender channel.	Enter option 14 (Start) next to TEST4_TO_1. TCP Sender channel.
Refresh the list periodically using function key 5 (PF5) until the channel TEST1_TO_4.TCP shows a status of RUNNING. The channel status continues to show STOPPED for a time, then shows BINDING followed by READY. The channel status continues to show STOPPED for a time. If the status does not show BINDING followed by READY, issue the command WRKSPLF QMQM and look at displayed list for a AMQRMCLA or AMQMCCLA joblog. The channel status continues to show STOPPED for a time. Display this joblog – somewhere near the end of the list there will be an error indicating the reason for the failure of the channel to start. Correct the error and retry the operation.	Refresh the list periodically using function key 5 (PF5) until the channel TEST4_TO_1.TCP shows a status of RUNNING. The channel status continues to show STOPPED for a time, then shows BINDING followed by READY. The channel status continues to show STOPPED for a time. If the status does not show BINDING followed by READY, issue the command WRKSPLF QMQM and look at displayed list for a AMQRMCLA or AMQMCCLA joblog. The channel status continues to show STOPPED for a time. Display this joblog – somewhere near the end of the list there will be an error indicating the reason for the failure of the channel to start. Correct the error and retry the operation.
Start the Administration utility by issuing the STRMQMADM command.	
Add the two systems TEST1 and TEST4, if you have not already done so, by using function key 6 (PF6) from the MQSeries Administration main menu.	
On the MQSeries Administration main menu select option 30 (Ping) against each queue manager and press the Enter key.	
On the MQSeries Administration main menu, use function key 10 (F10 – Responses) to view the results.	

Table 7 (Page 2 of 2). Queue manager link verification

On System TEST1	On System TEST4
The status of the channel can be displayed by working with the WRKMQMCHST command.	

Checking queue statistics

To check statistics on a queue you carry out the following procedure in the order shown:

Table 8. Queue statistics verification

Panel	Actions
`MQSeries Administration`	Type `8` in the `Opt` column next to the message queue manager that you require and press the `Enter` key.
`MQSeries Object Types`	Type in the specific queue name, or generic name in the `Object Name` field. Enter a forward slash (/) character beside all the queue types and press the `Enter` key. Wait for the information requested from the selected queue manager. A status message is displayed to inform you what is happening.
`MQSeries Work with Objects`	Locate the queue in the list by scrolling, or using the `Position to` function (`F17`).
`MQSeries Work with Objects`	Type 12 in the `Opt` column next to the selected queue and press the `Enter` key.
`MQSeries Work with Objects`	Press the `Responses` key (`F10`).
`MQSeries Work with Responses`	Locate the "Reset Q Statistics" line. If the `Avl` field is `N`, press the `Refresh` key (`F5`) periodically until `Y` appears, or check that the command server is running on the selected message queue manager. If the `Avl` field is `Y`, type `8` in the `Opt` column and press the `Enter` key.
`MQSeries Display Response`	The following details are displayed: `Q NAME` — The name of the queue manager name. `MSG ENQ COUNT` — The number of messages put on to the queue. `MSG DEQ COUNT` — The number of messages got from the queue. `HIGH Q DEPTH` — The largest number of messages on the queue. `TIME SINCE RESET` — The number of seconds since the last Reset was performed. The values are set to zero by this task. An example of this panel is shown in Figure 20 on page 61.

Appendix C. Specific MQSeries for AS/400 considerations

This appendix contains information about specific restrictions that are applicable to certain MQSeries for AS/400 objects.

AS/400 commands accessed by QMQM

```
/*******************************************************************/
/* Modify some of the authorities to let QMQM do its job           */
/*******************************************************************/
            /* let QMQM put jobs on the QSYSNOMAX job queue */
            QSYS/GRTOBJAUT OBJ(QSYS/QSYSNOMAX) OBJTYPE(*JOBQ) +
                         USER(QMQM) AUT(*USE)

            /* let QMQM use the CHGJRN command              */
            QSYS/GRTOBJAUT OBJ(QSYS/CHGJRN) OBJTYPE(*CMD) +
                         USER(QMQM) AUT(*USE)

            /* let QMQM acces the QPGMR user profile        */
            QSYS/GRTOBJAUT OBJ(QSYS/QPGMR) OBJTYPE(*USRPRF) +
                         USER(QMQM) AUT(*CHANGE)
```

MQSeries for AS/400 commands having restricted usage

Table 9 contains details of commands that have restricted usage. If the command is not listed it is available to all users.

Table 9 (Page 1 of 2). Command authority

Home libraries	Command name	Authority	Users
QMQM	RCDMQMIMG	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	RCRMQMOBJ	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	CHGMQMCHL	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	CPYMQMCHL	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	CRTMQMCHL	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	DLTMQMCHL	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	ENDMQMCHL	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	RSTMQMCHL	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	RSVMQMCHL	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	STRMQMCHL	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	STRMQMCHLI	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC

© Copyright IBM Corp. 1994, 1998

Table 9 (Page 2 of 2). Command authority

Home libraries	Command name	Authority	Users
QMQM	STRMQMLSR	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	CRTMQM	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	DLTMQM	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	STRMQM	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	ENDMQM	*USE	QSYSOPR QPGMR
		*EXCLUDE	*PUBLIC
QMQM	STRMQMSRV	*USE	QSYSOPR QPGMR QSRV QSRVBAS QRJE
		*EXCLUDE	*PUBLIC
QMQM	ENDMQMSRV	*USE	QSYSOPR QPGMR QSRV QSRVBAS QRJE
		*EXCLUDE	*PUBLIC
QMQM	TRCMQM	*USE	QSYSOPR QPGMR QSRV QSRVBAS QRJE
		*EXCLUDE	*PUBLIC
QMQM	all others	*USE	*PUBLIC

Shipped programs

The following programs are shipped with MQSeries for AS/400.

Program	`QMQM/AMQSDEF4`
Use	Creates the system default queues and channels
Parameters	None
Public authority	*USE
Further information	Chapter 4, “Operating MQSeries for AS/400 using CL commands” on page 31, Chapter 9, “Commands and utilities” on page 101, and “AMQSDEF4” on page 241

Program	`QMQM/AMQSADM4`
Use	Sets up the environment for you to use the Adminstration utility
Parameters	User ID
Public authority	*USE
Further information	Chapter 6, “Recovery” on page 63, Chapter 9, “Commands and utilities” on page 101, and “AMQSADM4” on page 253

Program	`QMQM/AMQIQEJ4`
Use	Ends all jobs owned by QMQM, temporarily revokes the object authorities to the QMQM and QMQMADM libraries, and performs a Record Object Image process.
Parameters	None
Public authority	*EXCLUDE
Further information	“Quiescing MQSeries for AS/400” on page 16

Program	`QMQM/AMQIQEM4`
Use	Tidies up the cache storage.
Parameters	None
Public authority	*EXCLUDE
Further information	“Quiescing MQSeries for AS/400” on page 16

Program	QMQM/AMQIQES4
Use	Ends all MQSeries jobs and revokes *PUBLIC and *USE authorities to the QMQM and QMQMADM libraries.
Parameters	None
Public authority	*EXCLUDE
Further information	“Quiescing MQSeries for AS/400” on page 16

Objects created by MQSeries for AS/400

Whenever you start any job, the system creates a QTEMP library, which is then deleted at the end of the job.

When the job is connected to the message queue manager, the following objects of type *USRSPC may also exist:

QMQMLOCAL	Created when the application connects to the message queue manager.
QAMGUS	Created when the display authority command is used.
QXCSGLOBAL	Created when any function of the message queue manager is used.
QFZRIOHEAP	Created when any function of the message queue manager is used.
QXERRTEMP	Created when any function of the message queue manager is used.

After disconnecting from the message queue manager, the following objects still exist:

QMQMLOCAL

QFZRIOHEAP

Note: QMQMLOCAL, QMQAGUS, and QXCSGLOBAL are owned by the message queue manager. The user must have the requisite authority to the message queue manager for these objects to be displayed at the terminal.

Jobs created by MQSeries for AS/400

Separately running MQSeries for AS/400 jobs are started in subsystem QSYSWRK by user QMQM. The jobs that run include:

Storage monitor (**AMQXIHK4**)	Starts when the first MQSeries for AS/400 operation is performed. In normal operation this job never ends.
Check point process (**AMQALMP4**)	Runs whenever the MQSeries for AS/400 queue manager is running.
Command server (**AMQPCSVA**)	Starts when the command STRMQMCSVR is issued, and ends when the command ENDMQMCSVR is issued.
Cache for Administration utility (**AMQMCPRA**)	Starts when the command STRMQMADM is issued. In normal operation this job never ends.
TCP/IP listener program (**AMQCLMAA**)	One listener program is started when the command STRMQMLSR is issued. The program monitors for incoming TCP/IP connections.

Channel initiator (**AMQRIMNA**)	Program activated for incoming TCP/IP channel connections.
Channel receiver, TCP/IP (**AMQCCCLA**)	Program acts as trigger monitor for sender MCA.
Channel receiver, LU 6.2 (**AMQCRS6A**)	Program activated for incoming LU 6.2 channel connections.
Channel MCA program (**AMQRMCLA**)	Starts when the command STRMQMCHL is issued. There is one MCA program for each channel started.

During normal operations you do not have to explicitly start or end these jobs because MQSeries for AS/400 starts and ends them when required by the appropriate MQSeries for AS/400 command.

When re-installing or deleting MQSeries for AS/400 any MQSeries for AS/400 jobs still running will be ended by quiescing MQSeries for AS/400 (see “Quiescing MQSeries for AS/400” on page 16).

Operating considerations

MQSeries for AS/400 runs in its own named activation group QMQM. Running the reclaim activation group command on the QMQM activation group, for example, issuing `RCLACTGRP ACTGRP(QMQM)`, will remove your queue manager connection handle.

MQSeries for AS/400 V4R2M1 commitment control (syncpointing) considerations

In order to participate in AS/400 units of work, MQSeries for AS/400 registers itself as an API commitment resource. This has a bearing on when you can end commitment control with the ENDCMTCTL command.

An overview of how MQSeries for AS/400 participates in current units of work is given in the *MQSeries Application Programming Guide*. Commitment definitions and API commitment resources are defined in the *AS/400 API Reference AS/400 API Programming*

Appendix D. MQSeries for AS/400 V4R2M1 at a glance

Program Number

| • 5769-MQ2 MQSeries for AS/400 Version 4 Release 2 Modification 1

Machine requirements

MQSeries for AS/400 runs on any AS/400 Advanced Series processor capable of running the required level of OS/400 and which has enough processor storage to meet the combined requirements of the programming prerequisites, the access methods, and the application programs.

Storage requirements for MQSeries for AS/400 V4R2M1

To install the product, MQSeries for AS/400 requires approximately 21 MB of storage, and in addition, you should allow a minimum of 40 MB when running the product.

Programming requirements

| The OS/400 V4R2 or V4R3 operating system is a prerequisite.

Connectivity can be through SNA LU 6.2 or TCP/IP.

Compilers supported for MQSeries for AS/400 applications

- IBM ILE C/400 Compiler (5769-CX2)
- IBM ILE COBOL/400 Compiler Version 3 (5769-CB1)
- IBM ILE RPG/400 Compiler (5769-RG1)
- IBM VisualAge C++ Compiler (5769-CX4)

Delivery

MQSeries for AS/400 is supplied on CD-ROM.

Installation

| MQSeries for AS/400 is installed with the AS/400 RSTLICPGM command.

The installation can be performed in approximately 10 minutes on an AS/400 model 510. Set up of the product to your specific requirements may then be required, the duration of this process being dependent on your specific needs.

© Copyright IBM Corp. 1994, 1998

Appendix E. Notices

The following paragraph does not apply to any country where such provisions are inconsistent with local law:INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact Laboratory Counsel, MP151, IBM United Kingdom Laboratories, Hursley Park, Winchester, Hampshire, England SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

Trademarks

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

AIX	Application System/400	AS/400
BookManager	C/400	CICS
COBOL/400	FFST	First Failure Support Technology
IBM	IMS	MQSeries
MQSeries Three Tier	MVS/ESA	OS/2
OS/400	RACF	VisualAge

Lotus and Lotus Notes, are registered trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation.

© Copyright IBM Corp. 1994, 1998

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

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

Glossary of terms and abbreviations

This glossary defines MQSeries terms and abbreviations used in this book. If you do not find the term you are looking for, see the Index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

This glossary includes terms and definitions from the *American National Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42 Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.

A

adapter. An interface between MQSeries for MVS/ESA and TSO, IMS, CICS, or batch address spaces. An adapter is an attachment facility that enables applications to access MQSeries services.

administrator commands. MQSeries commands used to manage MQSeries objects, such as queues, processes, and namelists.

alert. A message sent to a management services focal point in a network to identify a problem or an impending problem.

alias queue object. An MQSeries object, the name of which is an alias for a base queue defined to the local queue manager. When an application or a queue manager uses an alias queue, the alias name is resolved and the requested operation is performed on the associated base queue.

alternate user security. A security feature in which the authority of one user ID can be used by another user ID; for example, to open an MQSeries object.

APAR. Authorized program analysis report.

application-defined format. In message queuing, application data in a message, which has a meaning defined by the user application. Contrast with *built-in format*.

application environment. The software facilities that are accessible by an application program. On the MVS platform, CICS and IMS are examples of application environments.

application queue. A queue used by an application.

asynchronous messaging. A method of communication between programs in which programs place messages on message queues. With asynchronous messaging, the sending program proceeds with its own processing without waiting for a reply to its message. Contrast with *synchronous messaging*.

attribute. One of a set of properties that defines the characteristics of an MQSeries object.

authorization checks. Security checks that are performed when a user tries to open an MQSeries object.

authorization file. In MQSeries on UNIX systems, a file that provides security definitions for an object, a class of objects, or all classes of objects.

authorized program analysis report (APAR). A report of a problem caused by a suspected defect in a current, unaltered release of a program.

B

back out. An operation that reverses all the changes made during the current unit of recovery or unit of work. After the operation is complete, a new unit of recovery or unit of work begins. Contrast with *commit*.

browse. In message queuing, to use the MQGET call to copy a message without removing it from the queue. See also *get*.

browse cursor. In message queuing, an indicator used when browsing a queue to identify the message that is next in sequence.

built-in format. In message queuing, application data in a message, which has a meaning defined by the queue manager. Synonymous with *in-built format*. Contrast with *application-defined format*.

C

call back. In MQSeries, a requester message channel initiates a transfer from a sender channel by first calling the sender, then closing down and awaiting a call back.

CCF. Channel control function.

CCSID. Coded character set identifier.

CDF. Channel definition file.

channel. See *message channel*.

© Copyright IBM Corp. 1994, 1998

channel control function (CCF). In MQSeries, a program to move messages from a transmission queue to a communication link, and from a communication link to a local queue, together with an operator panel interface to allow the setup and control of channels.

channel definition file (CDF). In MQSeries, a file containing communication channel definitions that associate transmission queues with communication links.

channel event. An event indicating that a channel instance has become available or unavailable. Channel events are generated on the queue managers at both ends of the channel.

channel exit program. A user-written program that can be entered from one of a defined number of places during channel operation.

channel initiator. A component of MQSeries distributed queuing, which monitors the initiation queue to see when triggering criteria have been met and then starts the sender channel.

channel listener. A component of MQSeries distributed queuing, which monitors the network for a startup request and then starts the receiving channel.

checkpoint. A time when significant information is written on the log. Contrast with *syncpoint*.

CI. Control interval.

CICS transaction. In CICS, a unit of application processing, usually comprising one or more units of work.

CL. Control Language.

client. A run-time component that provides access to queuing services on a server for local user applications. The queues used by the applications reside on the server. See also *MQSeries client*.

client application. An application, running on a workstation and linked to a client, that gives the application access to queuing services on a server.

client connection channel type. The type of MQI channel definition associated with an MQSeries client. See also *server connection channel type*.

coded character set identifier (CCSID). The name of a coded set of characters and their code point assignments.

command. In MQSeries, an instruction that can be carried out by the queue manager.

command processor. The MQSeries component that processes commands.

command server. The MQSeries component that reads commands from the system-command input queue, verifies them, and passes valid commands to the command processor.

commit. An operation that applies all the changes made during the current unit of recovery or unit of work. After the operation is complete, a new unit of recovery or unit of work begins. Contrast with *backout*.

completion code. A return code indicating how an MQI call has ended.

connect. To provide a queue manager connection handle, which an application uses on subsequent MQI calls. The connection is made either by the MQCONN call, or automatically by the MQOPEN call.

connection handle. The identifier or token by which a program accesses the queue manager to which it is connected.

context. Information about the origin of a message.

context security. In MQSeries, a method of allowing security to be handled such that messages are obliged to carry details of their origins in the message descriptor.

Control Language (CL). In MQSeries for AS/400, a language that can be used to issue commands, either at the command line or by writing a CL program.

controlled shutdown. See *quiesced shutdown*.

D

data conversion interface (DCI). The MQSeries interface to which customer- or vendor-written programs that convert application data between different machine encodings and CCSIDs must conform. A part of the MQSeries Framework.

datagram. The simplest message that MQSeries supports. This type of message does not require a reply.

DCE. Distributed Computing Environment.

DCE principal. A user ID that uses the distributed computing environment.

DCI. Data conversion interface.

data-conversion service. A service that converts application data to the character set and encoding that are required by applications on other platforms.

dead-letter queue (DLQ). A queue to which a queue manager or application sends messages that it cannot deliver to their correct destination.

dead-letter queue handler. An MQSeries-supplied utility that monitors a dead-letter queue (DLQ) and processes messages on the queue in accordance with a user-written rules table.

default object. A definition of an object (for example, a queue) with all attributes defined. If a user defines an object but does not specify all possible attributes for that object, the queue manager uses default attributes in place of any that were not specified.

distributed application. In message queuing, a set of application programs that can each be connected to a different queue manager, but that collectively constitute a single application.

Distributed Computing Environment (DCE). Middleware that provides some basic services, making the development of distributed applications easier. DCE is defined by the Open Software Foundation (OSF).

distributed queue management (DQM). In message queuing, the setup and control of message channels to queue managers on other systems.

DLQ. Dead-letter queue.

DQM. Distributed queue management.

dynamic queue. A local queue created when a program opens a model queue object. See also *permanent dynamic queue* and *temporary dynamic queue*.

E

environment. See *application environment*.

event. See *channel event*, *instrumentation event*, *performance event*, and *queue manager event*.

event data. In an event message, the part of the message data that contains information about the event (such as the queue manager name, and the application that gave rise to the event). See also *event header*.

event header. In an event message, the part of the message data that identifies the event type of the reason code for the event.

event message. Contains information (such as the category of event, the name of the application that caused the event, and queue manager statistics) relating to the origin of an instrumentation event in a network of MQSeries systems.

event queue. The queue onto which the queue manager puts an event message after it detects an event. Each category of event (queue manager, performance, or channel event) has its own event queue.

F

FAP. Formats and Protocols.

FFST. First Failure Support Technology.

FIFO. First-in-first-out.

First Failure Support Technology (FFST). Used by MQSeries on UNIX systems, MQSeries for OS/2, MQSeries for Windows NT, and MQSeries for AS/400 to detect and report software problems.

first-in-first-out (FIFO). A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time. (A)

format. In message queuing, a term used to identify the nature of application data in a message. See also *built-in format* and *application-defined format*.

Formats and Protocols (FAP). The MQSeries FAPs define how queue managers communicate with one another, and also how MQSeries clients communicate with server queue managers.

Framework. In MQSeries, a collection of programming interfaces that allow customers or vendors to write programs that extend or replace certain functions provided in MQSeries products. The interfaces are:

- MQSeries data conversion interface (DCI)
- MQSeries message channel interface (MCI)
- MQSeries name service interface (NSI)
- MQSeries security enabling interface (SEI)
- MQSeries trigger monitor interface (TMI)

G

get. In message queuing, to use the MQGET call to remove a message from a queue. See also *browse*.

H

handle. See *connection handle* and *object handle*.

heartbeat flow. A pulse that is passed from a sending MCA to a receiving MCA when there are no messages to send. The pulse unblocks the receiving MCA, which otherwise, would remain in a wait state until a message arrived or the disconnect interval expired.

heartbeat interval. The time, in seconds, that is to elapse between heartbeat flows.

I

immediate shutdown. In MQSeries, a shutdown of a queue manager that does not wait for applications to disconnect. Current MQI calls are allowed to complete, but new MQI calls fail after an immediate shutdown has been requested. Contrast with *quiesced shutdown* and *preemptive shutdown*.

in-built format. See *built-in format*.

initialization file. In MQSeries for AS/400, a file that is used to customize channels for distributed queuing operations. This file is called QMINI. A default version is created, when MQSeries is installed, in library QMQMDATA.

initiation queue. A local queue on which the queue manager puts trigger messages.

input/output parameter. A parameter of an MQI call in which you supply information when you make the call, and in which the queue manager changes the information when the call completes or fails.

input parameter. A parameter of an MQI call in which you supply information when you make the call.

instrumentation event. A facility that can be used to monitor the operation of queue managers in a network of MQSeries systems. MQSeries provides instrumentation events for monitoring queue manager resource definitions, performance conditions, and channel conditions. Instrumentation events can be used by a user-written reporting mechanism in an administration application that displays the events to a system operator. They also allow applications acting as agents for other administration networks to monitor reports and create the appropriate alerts.

IPCS. Interactive Problem Control System.

L

listener. In MQSeries distributed queuing, a program that monitors for incoming network connections.

local definition. An MQSeries object belonging to a local queue manager.

local definition of a remote queue. An MQSeries object belonging to a local queue manager. This object defines the attributes of a queue that is owned by another queue manager. In addition, it is used for queue-manager aliasing and reply-to-queue aliasing.

local queue. A queue that belongs to the local queue manager. A local queue can contain a list of messages waiting to be processed. Contrast with *remote queue*.

local queue manager. The queue manager to which a program is connected and that provides message queuing services to the program. Queue managers to which a program is not connected are called *remote queue managers*, even if they are running on the same system as the program.

log. In MQSeries, a file recording the work done by queue managers while they receive, transmit, and deliver messages.

M

message. In message queuing applications, a communication sent between programs. See also *persistent message* and *nonpersistent message*. In system programming, information intended for the terminal operator or system administrator.

message channel. In distributed message queuing, a mechanism for moving messages from one queue manager to another. A message channel comprises two message channel agents (a sender and a receiver) and a communication link. Contrast with *MQI channel*.

message channel agent (MCA). A program that transmits prepared messages from a transmission queue to a communication link, or from a communication link to a destination queue.

message channel interface (MCI). The MQSeries interface to which customer- or vendor-written programs that transmit messages between an MQSeries queue manager and another messaging system must conform. A part of the MQSeries Framework.

message descriptor. Control information describing the message format and presentation that is carried as part of an MQSeries message. The format of the message descriptor is defined by the MQMD structure.

message priority. In MQSeries, an attribute of a message that can affect the order in which messages on a queue are retrieved, and whether a trigger event is generated.

message queue. Synonym for *queue*.

message queue interface (MQI). The programming interface provided by the MQSeries queue managers. This programming interface allows application programs to access message queuing services.

message queuing. A programming technique in which each program within an application communicates with the other programs by putting messages on queues.

message-retry. An option available to an MCA that is unable to deliver a message. The MCA can wait for a predefined amount of time and then try to send the message again.

message sequence numbering. A programming technique in which messages are given unique numbers during transmission over a communication link. This enables the receiving process to check whether all messages are received, to place them in a queue in the original order, and to discard duplicate messages.

messaging. See *synchronous messaging* and *asynchronous messaging*.

model queue object. A set of queue attributes that act as a template when a program creates a dynamic queue.

MQI. Message queue interface.

MQI channel. Connects an MQSeries client to a queue manager on a server system, and transfers only MQI calls and responses in a bidirectional manner. Contrast with *message channel*.

MQSC. MQSeries commands.

MQSeries. A family of IBM licensed programs that provides message queuing services.

MQSeries client. Part of an MQSeries product that can be installed on a system without installing the full queue manager. The MQSeries client accepts MQI calls from applications and communicates with a queue manager on a server system.

MQSeries commands (MQSC). Human readable commands, uniform across all platforms, that are used to manipulate MQSeries objects. Contrast with *programmable command format (PCF)*.

N

namelist. An MQSeries for MVS/ESA object that contains a list of queue names.

nonpersistent message. A message that does not survive a restart of the queue manager. Contrast with *persistent message*.

null character. The character that is represented by `X'00'`.

O

OAM. Object authority manager.

object. In MQSeries, an object is a queue manager, a queue, a process definition, a channel, a namelist (MVS/ESA only), or a storage class (MVS/ESA only).

object descriptor. A data structure that identifies a particular MQSeries object. Included in the descriptor are the name of the object and the object type.

object handle. The identifier or token by which a program accesses the MQSeries object with which it is working.

output parameter. A parameter of an MQI call in which the queue manager returns information when the call completes or fails.

P

PCF. Programmable command format.

PCF command. See *programmable command format*.

percolation. In error recovery, the passing along a preestablished path of control from a recovery routine to a higher-level recovery routine.

performance event. A category of event indicating that a limit condition has occurred.

performance trace. An MQSeries trace option where the trace data is to be used for performance analysis and tuning.

permanent dynamic queue. A dynamic queue that is deleted when it is closed only if deletion is explicitly requested. Permanent dynamic queues are recovered if the queue manager fails, so they can contain persistent messages. Contrast with *temporary dynamic queue*.

persistent message. A message that survives a restart of the queue manager. Contrast with *nonpersistent message*.

ping. In distributed queuing, a diagnostic aid that uses the exchange of a test message to confirm that a message channel or a TCP/IP connection is functioning.

platform. In MQSeries, the operating system under which a queue manager is running.

preemptive shutdown. In MQSeries, a shutdown of a queue manager that does not wait for connected applications to disconnect, nor for current MQI calls to complete. Contrast with *immediate shutdown* and *quiesced shutdown*.

process definition object. An MQSeries object that contains the definition of an MQSeries application. For example, a queue manager uses the definition when it works with trigger messages.

programmable command format (PCF). A type of MQSeries message used by:

- User administration applications, to put PCF commands onto the system command input queue of a specified queue manager
- User administration applications, to get the results of a PCF command from a specified queue manager
- A queue manager, as a notification that an event has occurred

Contrast with *MQSC*.

program temporary fix (PTF). A solution or by-pass of a problem diagnosed by IBM field engineering as the result of a defect in a current, unaltered release of a program.

PTF. Program temporary fix.

Q

queue. An MQSeries object. Message queuing applications can put messages on, and get messages from, a queue. A queue is owned and maintained by a queue manager. Local queues can contain a list of messages waiting to be processed. Queues of other types cannot contain messages—they point to other queues, or can be used as models for dynamic queues.

queue manager. A system program that provides queuing services to applications. It provides an application programming interface so that programs can access messages on the queues that the queue manager owns. See also *local queue manager* and *remote queue manager*. An MQSeries object that defines the attributes of a particular queue manager.

queue manager event. An event that indicates:

- An error condition has occurred in relation to the resources used by a queue manager. For example, a queue is unavailable.
- A significant change has occurred in the queue manager. For example, a queue manager has stopped or started.

queuing. See *message queuing*.

quiesced shutdown. In MQSeries, a shutdown of a queue manager that allows all connected applications to disconnect. Contrast with *immediate shutdown* and *preemptive shutdown*. A type of shutdown of the CICS adapter where the adapter disconnects from MQSeries, but only after all the currently active tasks have been completed.

quiescing. In MQSeries, the state of a queue manager prior to it being stopped. In this state, programs are allowed to finish processing, but no new programs are allowed to start.

R

reason code. A return code that describes the reason for the failure or partial success of an MQI call.

receiver channel. In message queuing, a channel that responds to a sender channel, takes messages from a communication link, and puts them on a local queue.

remote queue. A queue belonging to a remote queue manager. Programs can put messages on remote queues, but they cannot get messages from remote queues. Contrast with *local queue*.

remote queue manager. To a program, a queue manager that is not the one to which the program is connected.

remote queue object. See *local definition of a remote queue*.

remote queuing. In message queuing, the provision of services to enable applications to put messages on queues belonging to other queue managers.

reply message. A type of message used for replies to request messages.

reply-to queue. The name of a queue to which the program that issued an MQPUT call wants a reply message or report message sent.

report message. A type of message that gives information about another message. A report message can indicate that a message has been delivered, has arrived at its destination, has expired, or could not be processed for some reason.

requester channel. In message queuing, a channel that may be started remotely by a sender channel. The requester channel accepts messages from the sender channel over a communication link and puts the messages on the local queue designated in the message. See also *server channel*.

request message. A type of message used to request a reply from another program.

resolution path. The set of queues that are opened when an application specifies an alias or a remote queue on input to an MQOPEN call.

resource. Any facility of the computing system or operating system required by a job or task.

resource manager. An application, program, or transaction that manages and controls access to shared resources such as memory buffers and data sets. MQSeries, CICS, and IMS are resource managers.

responder. In distributed queuing, a program that replies to network connection requests from another system.

resynch. In MQSeries, an option to direct a channel to start up and resolve any in-doubt status messages, but without restarting message transfer.

return codes. The collective name for completion codes and reason codes.

return-to-sender. An option available to an MCA that is unable to deliver a message. The MCA can send the message back to the originator.

rollback. Synonym for *back out*.

rules table. A control file containing one or more rules that the dead-letter queue handler applies to messages on the DLQ.

S

security enabling interface (SEI). The MQSeries interface to which customer- or vendor-written programs that check authorization, supply a user identifier, or perform authentication must conform. A part of the MQSeries Framework.

SEI. Security enabling interface.

sender channel. In message queuing, a channel that initiates transfers, removes messages from a transmission queue, and moves them over a communication link to a receiver or requester channel.

sequential delivery. In MQSeries, a method of transmitting messages with a sequence number so that the receiving channel can reestablish the message sequence when storing the messages. This is required where messages must be delivered only once, and in the correct order.

sequential number wrap value. In MQSeries, a method of ensuring that both ends of a communication link reset their current message sequence numbers at the same time. Transmitting messages with a sequence number ensures that the receiving channel can reestablish the message sequence when storing the messages.

server. (1) In MQSeries, a queue manager that provides queue services to client applications running on a remote workstation. (2) The program that responds to requests for information in the particular two-program, information-flow model of client/server. See also *client*.

server channel. In message queuing, a channel that responds to a requester channel, removes messages from a transmission queue, and moves them over a communication link to the requester channel.

server connection channel type. The type of MQI channel definition associated with the server that runs a queue manager. See also *client connection channel type*.

service interval. A time interval, against which the elapsed time between a put or a get and a subsequent get is compared by the queue manager in deciding whether the conditions for a service interval event have been met. The service interval for a queue is specified by a queue attribute.

service interval event. An event related to the service interval.

shutdown. See *immediate shutdown*, *preemptive shutdown*, and *quiesced shutdown*.

single-phase backout. A method in which an action in progress must not be allowed to finish, and all changes that are part of that action must be undone.

single-phase commit. A method in which a program can commit updates to a queue without coordinating those updates with updates the program has made to resources controlled by another resource manager. Contrast with *two-phase commit*.

SNA. Systems network architecture.

source queue manager. See *local queue manager.*

star-connected communications network. A network in which all nodes are connected to a central node.

store and forward. The temporary storing of packets, messages, or frames in a data network before they are retransmitted toward their destination.

symptom string. Diagnostic information displayed in a structured format designed for searching the IBM software support database.

synchronous messaging. A method of communication between programs in which programs place messages on message queues. With synchronous messaging, the sending program waits for a reply to its message before resuming its own processing. Contrast with *asynchronous messaging*.

syncpoint. An intermediate or end point during processing of a transaction at which the transaction's protected resources are consistent. At a syncpoint, changes to the resources can safely be committed, or they can be backed out to the previous syncpoint.

system.command.input queue. A local queue on which application programs can put MQSeries commands. The commands are retrieved from the queue by the command server, which validates them and passes them to the command processor to be run.

system control commands. Commands used to manipulate platform-specific entities such as buffer pools, storage classes, and page sets.

systems network architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks.

T

target queue manager. See *remote queue manager*.

temporary dynamic queue. A dynamic queue that is deleted when it is closed. Temporary dynamic queues are not recovered if the queue manager fails, so they can contain nonpersistent messages only. Contrast with *permanent dynamic queue*.

thread. In MQSeries, the lowest level of parallel execution available on an operating system platform.

time-independent messaging. See *asynchronous messaging*.

TMI. Trigger monitor interface.

trace. In MQSeries, a facility for recording MQSeries activity. The destinations for trace entries can include GTF and the system management facility (SMF). See *performance trace*.

transaction. See *unit of work* and *CICS transaction*.

transaction manager. A software unit that coordinates the activities of resource managers by managing global transactions and coordinating the decision to commit them or roll them back. V5.0 of MQSeries for AIX, HP-UX, OS/2, Sun Solaris, and Windows NT is a transaction manager.

transmission program. See *message channel agent*.

transmission queue. A local queue on which prepared messages destined for a remote queue manager are temporarily stored.

trigger event. An event (such as a message arriving on a queue) that causes a queue manager to create a trigger message on an initiation queue.

triggering. In MQSeries, a facility allowing a queue manager to start an application automatically when predetermined conditions on a queue are satisfied.

trigger message. A message containing information about the program that a trigger monitor is to start.

trigger monitor. A continuously-running application serving one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the message. It uses the information in the trigger message to start a process that serves the queue on which a trigger event occurred.

trigger monitor interface (TMI). The MQSeries interface to which customer- or vendor-written trigger monitor programs must conform. A part of the MQSeries Framework.

two-phase commit. A protocol for the coordination of changes to recoverable resources when more than one resource manager is used by a single transaction. Contrast with *single-phase commit*.

U

UIS. User identifier service.

undelivered-message queue. See *dead-letter queue*.

unit of recovery. A recoverable sequence of operations within a single resource manager. Contrast with *unit of work*.

unit of work. A recoverable sequence of operations performed by an application between two points of consistency. A unit of work begins when a transaction starts or after a user-requested syncpoint. It ends either at a user-requested syncpoint or at the end of a transaction. Contrast with *unit of recovery*.

utility. In MQSeries, a supplied set of programs that provide the system operator or system administrator with facilities in addition to those provided by the MQSeries commands. Some utilities invoke more than one function.

X

X/Open XA. The X/Open Distributed Transaction Processing XA interface. A proposed standard for distributed transaction communication. The standard specifies a bidirectional interface between resource managers that provide access to shared resources within transactions, and between a transaction service that monitors and resolves transactions.

Index

Sending your comments to IBM

MQSeries for AS/400

Administration Guide

GC33-1956-01

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, use the Readers' Comment Form
- By fax:
  - From outside the U.K., after your international access code use 44 1962 870229
  - From within the U.K., use 01962 870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: WINVMD(IDRCF)
  - Internet: idrcf@winvmd.vnet.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Readers' Comments

MQSeries for AS/400

Administration Guide

GC33-1956-01

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

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

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Telephone	Email

MQSeries for AS/400

MQSeries for AS/400 V4R2.1 Administration Guide GC33-1956-01

IBM

You can send your comments POST FREE on this form from any one of these countries:

Australia	Finland	Iceland	Netherlands	Singapore	United States of America
Belgium	France	Israel	New Zealand	Spain	
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

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

1 Cut along this line

2 Fold along this line

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR

NO STAMP REQUIRED

IBM

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories
Information Development Department (MP095)
Hursley Park,
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name ____________________

Company or Organization ____________________

Address ____________________

____

EMAIL ____________________

Telephone ____________________

1 Cut along this line

4 Fasten here with adhesive tape

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC33-1956-01

Spine information:

IBM MQSeries for AS/400 Administration Guide *Version 4 Release 2.1*