MQSeries®

Planning Guide

GC33-1349-08

MQSeries®

Planning Guide

GC33-1349-08

Note!
Before using this information and the product it supports, be sure to read the general information under "Appendix. Notices" on page 265.

Ninth edition (March 2000)

This edition applies to the following products:
- MQSeries for AIX® Version 5 Release 1
- MQSeries for AS/400® Version 5 Release 1
- MQSeries for AT&T GIS UNIX® Version 2 Release 2
- MQSeries for Compaq (DIGITAL) OpenVMS Alpha Version 2 Release 2.1.1
- MQSeries for Compaq (DIGITAL) OpenVMS VAX Version 2 Release 2.1.1
- MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX) Version 2 Release 2.1 (PRPQ)
- MQSeries for HP-UX Version 5 Release 1
- MQSeries for OS/2® Warp Version 5 Release 1
- MQSeries for OS/390® Version 2 Release 1
- MQSeries for SINIX and DC/OSx Version 2 Release 2
- MQSeries for Solaris on Intel® Version 5 Release 0 (PRPQ)
- MQSeries for Sun Solaris Version 5 Release 1
- MQSeries for Tandem NonStop Kernel Version 2 Release 2.0.1
- MQSeries for VSE/ESA™ Version 2 Release 1
- MQSeries for Windows NT® Version 5 Release 1
- MQSeries for Windows NT on DIGITAL Alpha (AXP) Version 2 Release 0.1 (PRPQ)
- MQSeries for Windows® Version 2 Release 0
- MQSeries for Windows Version 2 Release 1
- MQSeries link for R/3 for AIX Version 1 Release 2
- MQSeries link for R/3 for AS/400 Version 1 Release 2
- MQSeries link for R/3 for DIGITAL UNIX (Compaq Tru64 UNIX) Version 1 Release 2 (PRPQ)
- MQSeries link for R/3 for HP-UX Version 1 Release 2
- MQSeries link for R/3 for Sun Solaris Version 1 Release 2
- MQSeries link for R/3 for Windows NT Version 1 Release 2

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

Contents

Figures

Tables

About this book

The MQSeries family of products provides application programming services that:

- Enable you to code indirect program-to-program communication using *message queues*
- Are independent of the platform, for example, OS/390 or OS/400®

This book explains the planning necessary for the incorporation of MQSeries products into your enterprise.

Who this book is for

This book is for:

- Planners of systems that will use MQSeries message-queuing techniques
- System programmers who have to install and customize MQSeries products for these systems

What you need to know to understand this book

To understand this book, you should be familiar with the system facilities for the platform on which you are installing the MQSeries product.

If you are unfamiliar with the concepts of messaging and queuing, you should read *An Introduction to Messaging and Queuing.*

Terms used in this book

This book uses the following shortened names:

MQSeries
General term for IBM MQSeries products.

CICS® General term for CICS and CICS Transaction Server on all platforms.

UNIX systems
General term referring to the following UNIX systems:

- AIX
- AT&T GIS UNIX [1]
- DIGITAL UNIX (Compaq Tru64 UNIX)
- HP-UX
- SINIX and DC/OSx
- Sun Solaris

1. This platform has become NCR UNIX SVR4 MP-RAS, R3.0.

Summary of changes

This section describes changes to this edition of the *MQSeries Planning Guide*. Changes made since the previous edition of the book was published are marked by vertical lines to the left of the changes.

Changes for this edition (GC33-1349-08)

This edition includes the following new and updated MQSeries products:

- MQSeries for AS/400, Version 5 Release 1
- MQSeries for Compaq (DIGITAL) OpenVMS, Version 2 Release 2.1.1
- MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX), Version 2 Release 2.1
- MQSeries link for R/3 for DIGITAL UNIX (Compaq Tru64 UNIX), Version 2 Release 2.1
- MQSeries for Tandem NonStop Kernel, Version 2 Release 2.0.1

Changes for the eighth edition

This edition, GC33-1349-07, includes the following new and updated MQSeries products:

- MQSeries for AIX, Version 5 Release 1
- MQSeries for AS/400, Version 4 Release 2.1
- MQSeries for HP-UX, Version 5 Release 1
- MQSeries for OS/2 Warp, Version 5 Release 1
- MQSeries for OS/390, Version 2 Release 1
- MQSeries for Sun Solaris, Version 5 Release 1
- MQSeries for VSE/ESA, Version 2 Release 1
- MQSeries for Windows NT, Version 5 Release 1
- MQSeries link for R/3 for AIX, Version 1.2
- MQSeries link for R/3 for AS/400, Version 1.2
- MQSeries link for R/3 for HP-UX, Version 1.2
- MQSeries link for R/3 for Sun Solaris, Version 1.2
- MQSeries link for R/3 for Windows NT, Version 1.2

Changes for the seventh edition

Changes for edition number GC33-1349-06 include:

- The book has been updated to contain information about the following new functions of the MQSeries for AS/400 Version 4 Release 2 product:
  - Distribution lists
  - Message segmentation
  - Fast channels for nonpersistent messages
  - Auto-definition of server-connection and receiver channels
  - Reference messages
  - C++ programming
- The book has been updated to contain revised information about MQSeries for Digital OpenVMS, Version 2 Release 2.

Changes

- The book has been updated to contain information about MQSeries for Tandem NSK, Version 2 Release 2.
- The book has been updated to contain information about the MQSeries client for VM/ESA®, Version 2 Release 3.

Part 1. Introduction to MQSeries

Introduction

Chapter 1. Introduction to IBM MQSeries

This chapter introduces IBM MQSeries and describes its relationship with other products. It contains basic explanations of the following topics:

For more detailed explanations of these topics, refer to the *MQSeries Application Programming Guide*.

MQSeries and message queuing

MQSeries products enable applications to use message queuing to participate in *message-driven processing*. With message-driven processing, applications can communicate with each other on the same or different platforms, by using the appropriate message queuing software products. For example, OS/390 and OS/400 applications can communicate through MQSeries for OS/390 and MQSeries for AS/400 respectively. With MQSeries products, all applications use the same kind of messages; communications protocols are hidden from the applications.

MQI – a common application programming interface

MQSeries products implement a common application programming interface, the *message queue interface* (MQI), that is used on whatever platform the applications run on. The calls made by the applications and the messages they exchange are common. This makes it much easier to write and maintain applications than using traditional methods. It also facilitates the migration of message queuing applications from one platform to another.

The MQI calls are shown in the following table.

Table 1. MQI calls

Call name	Description	Platforms
MQBEGIN	Begin a unit of work to be coordinated by the queue manager	AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT
MQCONN	Connect to a queue manager	All platforms
MQCONNX	Connect to a queue manager and specify some options	AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT
MQDISC	Disconnect from a queue manager	All platforms
MQOPEN	Open an object	All platforms
MQCLOSE	Close an object	All platforms
MQPUT	Put a message (queue already open)	All platforms
MQPUT1	Put one message	All platforms

Table 1. MQI calls (continued)

Call name	Description	Platforms
MQGET	Get a message	All platforms
MQCMIT	Commit changes	OS/2 Warp, OS/390, OS/400, Windows NT, and UNIX systems
MQBACK	Back out changes	OS/2 Warp, OS/390, OS/400, Windows NT, and UNIX systems
MQINQ	Inquire about object attributes	All platforms
MQSET	Set object attributes	All platforms

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. The receiving application might be busy when the message is sent. Indeed, the receiving application doesn't even need to be running. MQSeries holds the message in the queue until it can be processed.

Message-driven processing

Message-driven processing is a style of application design.

With this style, the application is divided into a number of separate, discrete, functional blocks, with each block having well-defined input and output parameters. Each functional block is coded as an application program, with its input and output parameters being interchanged between other application programs by placing their values in messages, which are then put on queues.

By use of the appropriate MQSeries programming mechanisms, an application program can start executing as a result of one or more messages arriving on a queue. If required, the program can terminate when all the messages in a queue have been processed.

This style of application design allows new applications to be built, or existing applications to be modified, more quickly than with some other application design styles.

Data integrity and resource protection

MQSeries applications can transfer data with an extremely high degree of confidence. Message delivery can be implemented using a syncpoint mechanism–and the MQSeries logs or journals–for the recovery of important data in the event of system failure. See "Chapter 4. Introduction to MQSeries recovery concepts" on page 27.

All resources, such as message queues, can be protected using the security facilities available on the operating platform; for example, Resource Access Control Facility (RACF®) on OS/390, or the security facilities provided by OS/400.

Messages and queues

Messages and queues are basic to any queuing system.

What is a message

A *message* is a string of bytes that has meaning to the applications that use the message.

In MQSeries, messages have two parts, a *message descriptor* and *application data*. The content and structure of the application data are defined by the application programs that use them. The message descriptor identifies the message and contains other control information or attributes, such as the date and time the message was created, the type of message, and the priority assigned to the message by the sending application.

Figure 1 represents a message and shows how the message is logically divided into message data and application data.

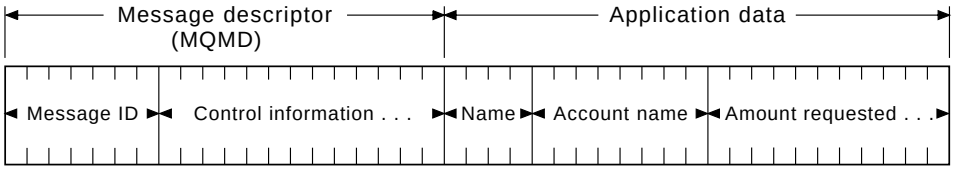

Figure 1. Representation of a message. The message descriptor and application data are shown as separate parts. Information that is specific to the application, such as Account name in this example, is in the application data part of the message.

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
- 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, in which case they are called *local queues*, or at another queue manager, in which case 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*.

Message attributes

For system administrators, messages have two important attributes that are defined in the message descriptor: *persistence* and *priority*.

A message is termed persistent if it survives when MQSeries restarts. This implies that the message must be logged, or saved, and can be reinstated as part of the recovery procedure.

Each MQSeries message has a priority assigned to it by the sending application. The priority, which is a number in the range 0 through 9, can affect the order in which a message is retrieved from a queue, and also the way that *trigger events* are generated.

Triggering is a facility in some MQSeries products. It allows an application to be started automatically when predefined conditions on a queue are met. These conditions include reception of any message or of messages over a particular priority, the number of messages on a queue, and so on. For more information about triggering, see the *MQSeries Application Programming Guide*.

Message sizes

The maximum message size supported by MQSeries varies. “MQSeries product functional comparison” on page 226 gives details of the size of message supported by each of the MQSeries products.

In practice, the size of message that an application program can put on a queue is limited by:

- The maximum message length defined for the receiving queue
- The maximum message length defined for the queue manager
- The maximum message length supported by the platform

If the size of the data that an application program requires to place on a queue exceeds this limit, the program must split the data into a number of pieces, and put each piece on the queue as a separate message.

Dealing with large messages

On some platforms, there are several methods available for dealing with very large messages.

Maximum message length

On AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT, you can use the maximum message length attribute to set the maximum length allowed for messages on a queue manager; this can be up to 100 MB. The maximum message length used for communication between two queue managers is the lower of their maximum message lengths, so you can use 100 MB messages only if both queue managers support them. The maximum size for queues on these platforms is 2 GB.

Message segmentation

On AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT, MQSeries allows you to divide large messages into several parts (that is, one logical message into several physical messages). Your application can define the size of physical messages, to allow for the maximum message size permitted at intermediate queue managers. MQSeries can start transmitting or receiving the physical messages before the whole of the logical message has arrived on the queue.

Reference messages

On AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT, reference messages allow you to transfer a large data object from one queue manager to another without storing the object on a queue at either the source or destination node. A message exit program is used to add the data object to the message when it is transmitted. At the receiving end, another message exit program is used to create a data object from the data in the message. The reference message, without the data, is then put on the destination queue. Sample exit programs that do this are provided with MQSeries.

Some queues used with MQSeries

MQSeries uses the following types of queues.

Message queues

A *message queue* is a queue that is used to receive messages from applications as distinct from those queues that have a special purpose. Special purpose queues are defined in the same way as message queues, although they can have their attributes set in specific ways. The system administrator is the person responsible for defining and maintaining all queues in your enterprise.

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. Instrumentation events help you to monitor your system. 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.

Initiation queues

An *initiation queue* receives *trigger messages*, which indicate that a trigger event has occurred. A trigger event is caused by a message that satisfies the specified conditions being put onto a queue. Messages are read from the initiation queue by a trigger monitor application which then starts the appropriate application to process the message. If triggers are active, at least one initiation queue must be defined for each queue manager.

Transmission queues

A *transmission queue* temporarily stores messages that are destined for a remote queue manager. You must define a transmission queue for each remote queue manager to which the local queue manager is to send messages. It is possible to associate several transmission queues with different characteristics with a remote queue manager. This allows different classes of transmission service.

Reply-to queues

If a message is a request message and so requires a reply, the sender of the message must specify the name of the queue to which the reply should be sent; this is called the *reply-to queue*. It is also the queue to which report messages are usually sent, if any are generated.

Dead-letter queues

A *dead-letter queue* (also known as an *undelivered-message queue*) receives messages that cannot be routed to their correct destinations. This occurs when, for example:

- The destination queue is full.
- The message cannot be put on the destination queue.
- The sender is not authorized to use the destination queue.
- The destination queue does not exist.

If you do not have a dead-letter queue, undelivered messages will remain on the transmission queue and the *message channel* will be stopped. You are therefore recommended to define a dead-letter queue for each queue manager in your system.

You also have to decide how to handle any message placed on a dead-letter queue. Such handling might be by the system administrator examining the queue and redirecting the message, or perhaps by an application which you create for the purpose of monitoring and handling dead-letter queues.

A dead-letter queue handler is provided with some MQSeries products to assist the system administrator with the task of dealing with messages placed on a dead-letter queue.

Command queues

A *command queue* is a queue owned by a queue manager to which suitably authorized applications can send messages containing MQSeries administration commands. The commands in these messages are processed by the *command server* part of the queue manager.

You can find more information on command queues in “Chapter 5. Introduction to MQSeries administration” on page 33.

Cluster queues

You can group queue managers together to form a cluster. The following queues are required to support clustering:

- A local queue is needed to hold a persistent copy of the repository. (The repository is a collection of information about the queue managers that are members of the cluster, held on some of the queue managers in the cluster and replicated on the other queue managers.) This queue is called SYSTEM.CLUSTER.REPOSITORY.QUEUE.
- A local queue is needed to communicate repository changes between queue managers. This queue is used for information about updates to the repository data to be applied to the local repository, or requests for repository data. This queue is called SYSTEM.CLUSTER.COMMAND.QUEUE.
- A local queue is needed as the transmission queue for all destinations in the cluster. This queue is called SYSTEM.CLUSTER.TRANSMIT.QUEUE.

System default queues

The *system default queues* are a set of queue definitions supplied with MQSeries. They are used to set default values for any attributes that you do not specify when defining your own queues. By modifying the supplied queue definitions you can vary the default queue attributes used at your installation.

MQSeries objects

An MQSeries object is a recoverable resource managed by MQSeries. Many of the tasks described in this book involve manipulating the following types of MQSeries object:

- Queue managers
- Queues
- Namelists
- Distribution lists
- Process definitions
- Channels
- Storage classes

These objects are common across different MQSeries platforms.

For system administrators, commands are available to manipulate objects. The format of the commands is dependent on the platform. For example, the MQSeries for OS/390 command DEFINE QLOCAL (with the appropriate attributes) defines a local queue object for MQSeries for OS/390. On MQSeries for AS/400, the equivalent command is CRTMQMQ.

On AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT, default objects are created for you when you create a queue manager. Default objects are supplied with the other MQSeries products, to help you define the objects that you need.

Each object has a *name* associated with it and can be referenced in MQSeries commands and MQI calls by that name. In general, 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. This means that you cannot have a local queue with the same name as a model queue, a remote queue, or an alias queue. There are a few exceptions to this rule; these are described where they arise.

Note: The characters within the names given to all MQSeries objects are case sensitive. Therefore, be very careful when defining the names of objects, to select the appropriate uppercase or lowercase characters.

Queue managers

A *queue manager* is that part of an MQSeries product that provides the messaging and queuing services to application programs, through the Message Queue Interface (MQI) program calls.

In some environments (for example, MQSeries for VSE/ESA) only one queue manager can be in use in one machine at any one time; in most environments, however, more than one queue manager can be executing in one machine at one time.

Introduction

In the OS/390 environment, applications are connected to a queue manager through adapters which are part of the MQSeries for OS/390 product. In the example shown in Figure 2, there are two applications on OS/390, one is a CICS transaction that is connected to the MQSeries queue manager by the CICS adapter, and the other a batch application connected to the queue manager by the batch adapter. Each application has access to local queues A1 and A2.

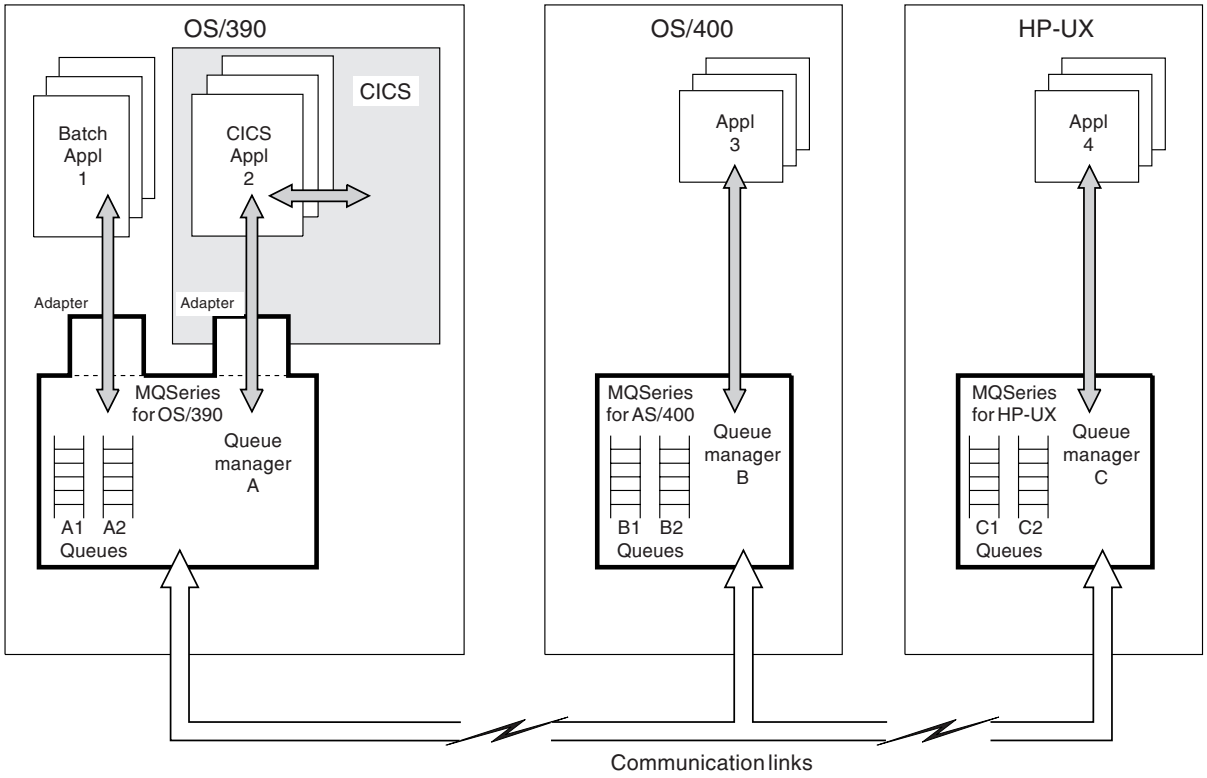

Figure 2. Example of messaging and queuing using MQSeries products

The applications issue the MQI calls that are implemented by the queue manager. For incoming messages, the queue manager directs them onto their respective destination queues; for outgoing messages, the queue manager sends the message to the destination queue manager. The destination queue manager ensures that the message is put onto the correct queue.

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

In Figure 2, queues A1 and A2 are local queues to both batch application 1, CICS application 2, and to any other application connected directly to queue manager A. The figure also illustrates remote queues B1 and B2 controlled by queue manager B, and C1 and C2 controlled by queue manager C. Application 3 can put messages either to its local queues B1 and B2 controlled by MQSeries for AS/400, or to remote queues A1, A2, C1, or C2, as it requires.

Queues

A queue is an MQSeries object that can store messages. Each queue has *queue attributes* that determine what happens when applications reference the queue in MQI calls. The attributes indicate:

- Whether applications can retrieve messages from the queue (get enabled)
- Whether applications can put messages onto 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 size of messages that can be put on the queue (maximum message size)

Using queue objects

In MQSeries, there are several types of queue object. This does not mean that there are several different types of queue; essentially there is only one type of queue. Each type of queue object can be manipulated by MQSeries commands and is associated with queues in different ways:

1. A *local queue object* defines a local queue belonging to the queue manager to which the application is connected.
2. A *remote queue object* identifies a queue belonging to another queue manager. The remote queue is usually given a local definition. The definition specifies the name of the remote queue manager where the queue exists as well as the name of the remote queue itself. The information you specify when you define a remote queue object enables 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* enables applications to access queues by referring to them indirectly in MQI calls. When an alias queue name is used in an MQI call, the name is resolved to the name of a message queue at run time. This enables you to change the queues that applications use without changing the application in any way; you merely change the alias definition.
4. The *model queue object* defines a set of queue attributes that are used as a template for a *dynamic queue*. Dynamic queues are created by the queue manager when an application makes an open queue request specifying a queue that is a model queue. The dynamic queue that is created in this way is a local queue whose attributes are those of the model queue. You can specify a name (in full) for the dynamic queue, or the stem of a name (for example, ABC) and let the queue manager add a unique part to this, or you can let the queue manager assign a complete unique name for you.

   Dynamic queues can be of two types:

   - *Temporary*

     This type of queue is deleted when the queue is closed, and does not survive a restart of a queue manager. The queues can be used to store nonpersistent messages only.

   - *Permanent*

     This type of queue is not deleted when the queue is closed, unless the application program specifically requests it to be deleted. The queue survives a restart of a queue manager, and can be used to store persistent and nonpersistent messages.

Note: Some MQSeries products do not support all the different types of queue object given above. Refer to "MQSeries product functional comparison" on page 226 for a summary of the functions offered on the different platforms.

Queue naming convention

If your applications operate within one local system, your application designers will specify the names of each queue they want to use. These queues will be named either to some local protocol or at the programmer's whim. In the latter case, it is immediately obvious that names chosen at random by individuals could be duplicates, causing all sorts of problems.

When your applications start to deal with remote queues, the need to know and understand a naming convention becomes essential. It is no good sending a message to a remote queue manager if the name of the desired queue is not known.

For all of the above reasons, it is recommended that you develop a naming convention for your enterprise which is known to, and understood by, all your application designers and developers. The end users of your applications probably do not need to know the convention because they will usually have no need to know the queue names.

Namelists

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

Note: Not all MQSeries products support namelists. Refer to "MQSeries product functional comparison" on page 226 for a summary of the functions offered on the different platforms.

Distribution lists

A *distribution list* provides a way for an application to send a message to several destinations with a single MQPUT call. The list of destinations is supplied by the application. If more than one of these destinations uses the same transmission queue, only one copy of the message data is kept on the transmission queue, and sent down the channel.

Note: Not all MQSeries products support distribution lists. Refer to "MQSeries product functional comparison" on page 226 for a summary of the functions offered on the different platforms.

Process definitions

A *process definition object* defines an application to an MQSeries queue manager. Typically, in MQSeries, an application puts or gets messages from one or more queues and processes them. A process definition object is used for defining applications to be started by a trigger monitor. The definition includes the application ID, the application type, and application specific data.

Trigger monitors

A *trigger monitor* is an application that monitors an initiation queue (see "Initiation queues" on page 7) associated with a queue manager. 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 on the initiation queue. For more information on triggering and trigger monitors see the *MQSeries Application Programming Guide*.

Note: Triggering is supported on MQSeries for VSE/ESA, but it does not use process objects. See the information about triggers in the *MQSeries for VSE/ESA System Management Guide* for more information about this.

Channels

A channel provides a communication path. There are two types of channel, message channels and MQI channels.

Message channels

A *message channel* provides a communication path between two queue managers on the same, or different, platforms. The message channel is used for the transmission of messages from one queue manager to another, and shields the application programs from the complexities of the underlying networking protocols.

A message channel can transmit messages in one direction only. If two-way communication is required between two queue managers, two message channels are required.

In order to set up a channel, you usually define one channel definition for each end of the channel. There are six types of message channel; Sender, Server, Receiver, Requester, Cluster-sender and Cluster-receiver. On some platforms, some types of message channel can be defined automatically.

MQI channels

An *MQI channel* connects an MQSeries client to a queue manager on a server machine. It is for the transfer of MQI calls and responses only and is bidirectional. A channel definition exists for each end of the link. There are two types of MQI channel; Server-connection and Client-connection. On some platforms, some types of MQI channel can be defined automatically.

Storage classes

A *storage class* is used on OS/390 to map one or more queues to a page set. See the *MQSeries for OS/390 System Management Guide* for more information about storage classes.

Clients and servers

An *MQSeries client* is a part of an MQSeries product that can be installed on a machine without installing the full queue manager. It accepts Message Queue Interface (MQI) calls from application programs, and passes MQI requests to an *MQSeries server* that is usually executing on another processor.

The *MQSeries server* is a full queue manager, which can accept MQI calls directly from application programs that are running on the server processor; in addition, it can accept MQI requests from MQSeries clients.

This allows you to have an application that uses the MQI running on one machine, the client machine, and the queue manager itself running on a different machine.

For further information, see "Chapter 7. Introduction to MQSeries clients and servers" on page 49.

Where to find more information

For an introduction to MQSeries on your platform, see the following chapters:

- "Chapter 9. Introduction to MQSeries for AS/400" on page 63
- "Chapter 14. Introduction to MQSeries for Compaq (DIGITAL) OpenVMS" on page 81
- "Chapter 19. Introduction to MQSeries for OS/2 Warp" on page 101
- "Chapter 24. Introduction to MQSeries for OS/390" on page 117
- "Chapter 32. Introduction to MQSeries for Tandem NSK" on page 155
- "Chapter 36. Introduction to MQSeries for UNIX systems" on page 167
- "Chapter 41. Introduction to MQSeries for VSE/ESA" on page 187
- "Chapter 42. Introduction to MQSeries for Windows NT" on page 193
- "Chapter 47. Introduction to MQSeries for Windows" on page 213

Chapter 2. Introduction to distributed queuing

The MQSeries distributed queuing facility moves messages between queue managers. To allow MQSeries to do this, you must ensure that you define and install the platform-dependent connections that provide the physical links between the local queue manager and any remote queue managers that you want to exchange messages with.

This chapter describes how to plan for communication between local and remote queue managers. It contains basic information about:

How queue managers communicate

There are two ways in which a queue manager can be connected to other queue managers. These queue managers might be on the same or different platforms. The connections can allow:

- Simple transfer
- Staged transfer

Simple transfer is used between two queue managers that are connected by an MQSeries message channel.

Staged transfer is used to interconnect queue managers that are located in nodes that are not adjacent to the sending node, and can only be reached by staging through an adjacent queue manager. A remote queue manager might find that some messages received are not for its local queues but, instead, are to be passed on to another queue manager that performs the same process of delivering the messages or, passing them on again.

In each of the models introduced above, you need to define transmission queues and links to the adjacent nodes. The adjacent, or neighboring, nodes on your network must also have queue managers available that can handle your messages. Such handling places the message on one of the receiving node's local queues when that node recognizes the queue name as one of its own queues, or forwards the message to another queue manager when a queue manager name other than its own is received.

You must ensure that neighboring nodes with which your system communicates either have links to the intended destination or, at least, links to the next step toward the intended destination.

The communication between two queue managers is by means of a message channel; this consists of:
- A *transmission queue*
- A *message channel agent* for each queue manager
- A *communications link*

A *message channel agent* is a program controlling communications and is part of the queue manager.

For a detailed description of how these components allow messages to be exchanged between systems, and how to define them, refer to the *MQSeries Intercommunication* manual.

Channel speed

For a normal channel, all messages travel through the message channel at the same speed. Persistent and nonpersistent messages on the same transmission queue maintain their order relative to each other. None of the messages are lost if there is a channel failure.

On AIX, HP-UX, OS/2 Warp, OS/390, OS/400, Sun Solaris, and Windows NT, you can define 'fast' channels. If a channel is defined to be fast, nonpersistent messages travel through the channel outside syncpoint. This improves the throughput of the channel, but means that nonpersistent messages are lost if there is a channel failure. (Persistent messages are never lost if a channel failure occurs.)

For a fast channel, it is possible for nonpersistent messages to jump ahead of persistent messages waiting on the same transmission queue, that is, the order of nonpersistent messages is not maintained relative to persistent messages. However the order of nonpersistent messages relative to each other is maintained. Similarly, the order of persistent messages relative to each other is maintained.

Setting up distributed queuing

To understand what you need to plan for, it is first necessary to review the queuing process.

When an application attempts to put a message to a queue, it must specify the destination queue. This is done by addressing the message to:
- An alias name
- A queue name, plus a queue manager name

If an alias name is used, the queue manager resolves that name to one of two destinations, either a local queue name, or the name of a remote queue. In the first case, the message is placed on the named local queue; in the second, the message is placed on the appropriate transmission queue.

In the case where the queue name, and the queue manager name are supplied, the queue manager name can be either that of the local queue manager, which is the default condition, or a remote queue manager name.

When a message is created by the queue manager as a result of an MQPUT call, the destination queue name, and queue manager name, are placed in the message descriptor part of the message.

When the queue manager sees that the queue manager name in a message is not its own name, it places the message on a special local queue called a transmission queue. From the transmission queue, the message is transferred to the next node in the network by the queue manager, using the message channel associated with that transmission queue. At the next node, the queue manager finds that the message is addressed to one of its local queues and accordingly places it on the destination queue, which can be another transmission queue giving further transmission to another node. You should note that there can be more than one transmission queue for each remote queue manager that you have defined. Refer to the *MQSeries Intercommunication* manual for more information about transmission queues.

A special case exists where a queue manager is called to handle a message that, though sent to, or defaulting to, this queue manager, does not have a known queue name. In this case the message is placed on the dead-letter queue. Special actions are required to recover or dispose of messages placed on the dead-letter queue. See "Dead-letter queues" on page 8 for more information.

To enable remote queuing, you might need to define remote queues. (This step is not essential; there are cases where the application can place the message directly to a queue manager and queue name combination.) The queue manager, when handling a request to place a message on a queue with either an alias name or a remote queue manager and queue name combination, uses the information to determine that the name refers to a remote queue manager. If so, the message is placed on a transmission queue associated with the remote queue manager. MQSeries then moves the message from the transmission queue to the remote queue manager.

You need to plan alias names for remote queues that you communicate with. You should consider the naming conventions that you wish to apply to these names, and also how you will ensure that your total network complies with your naming convention.

You must also plan the message *channels* that you will create for message transmission to remote queue managers. You might choose to define multiple channels to these remote queue managers to allow for high message traffic, different message priorities, or for different message types.

Application data conversion

The representation of character and numeric data is different on the various platforms for which MQSeries products are available. Because of this, you might need to plan for the conversion of the data within your messages, from one representation to another, when writing applications that span multiple platforms.

The message descriptor in all messages contains the coded character set identifier (CCSID) and encoding information; this identifies the representation used for the character and numeric data that is in the data portion of the message.

Some MQSeries products convert the data within messages from one representation to another, provided the format of the data conforms to one of the MQI built-in formats. This conversion can be performed at the following times:

- By a queue manager during the processing of an MQGET call, if the data conversion option is included in the call.

- By a message channel as it transmits a message to a remote queue manager, if the data conversion option is included in the channel definition. The channel converts the message data to the representation used by the platform at the receiving end of the channel.

For application defined formats that do not conform to the built-in formats, the conversion can be performed by user exit programs.

For further information about data conversion, refer to the *MQSeries Application Programming Guide*.

Assured delivery

MQSeries products are designed for assured message delivery. Processing is such that when messages are being transmitted to remote queue managers, the messages are moved in discrete transaction units, or batches, where confirmation of receipt is always obtained before a particular message is deleted at the transmitting queue manager. To achieve this, the sending and receiving ends of the link commit batches of messages in unison.

Recovering from errors

If a remote queuing error or session error occurs, error messages are sent to the local system operator or console, the local system being that where the queue manager detecting the error is running.

If any error occurs while sending or receiving a message, the transaction is terminated, and error messages are sent to both the local and remote system consoles. When you restart remote queuing, the queue manager checks for, and resolves, any in-doubt messages caused by the previous termination. An in-doubt message is one where the point of consistency prior to the message is known, but it is not known with certainty whether the new point of consistency that the message creates has been reached, or that a backout to a previous point of consistency has been completed. The application must decide what action to take to resolve the in-doubt situation.

If a message cannot be put on the queue on the remote queue manager, the message is written to the dead-letter queue on that queue manager and a report is sent back to the message sender (if so requested in the message).

Remember that it is important to ensure that there is a dead-letter queue defined for each queue manager, and that this queue is not allowed to fill completely. Otherwise, when an error is detected by the channel, the channel will stop transmission. See “Dead-letter queues” on page 8 for more information.

Using MQSeries clusters

As already discussed, every queue manager in a traditional MQSeries network is independent. If one queue manager needs to send messages to another it must define a transmission queue, a channel to the remote queue manager, and a remote queue definition for every queue to which it wants to send messages.

If you group queue managers in a cluster, the queue managers can make the queues that they host available to every other queue manager in the cluster. Then, assuming you have the necessary network infrastructure in place, any queue manager can send a message to any other queue manager in the same cluster without the need for explicit channel definitions, remote-queue definitions, or transmission queues for each destination. Every queue manager in a cluster has a single transmission queue that transmits messages to any other queue manager in the cluster. Each queue manager needs to define only one cluster-receiver channel on which to receive messages and one cluster-sender channel on which to send messages.

Figure 3 shows the components of a cluster called CLUSTER. In this cluster there are three queue managers, QM1, QM2, and QM3. QM1 and QM2 host repositories of information about the queue managers and queues in the cluster. QM2 and QM3 host some cluster queues, that is, queues that are accessible to any other queue manager in the cluster. Each queue manager has a cluster-receiver channel called TO.*qmgr* on which it can receive messages. Each queue manager also has a cluster-sender channel on which it can send cluster information and messages to one of the repository queue managers. In Figure 3, QM1 and QM3 send to the repository at QM2, and QM2 sends to the repository at QM1.

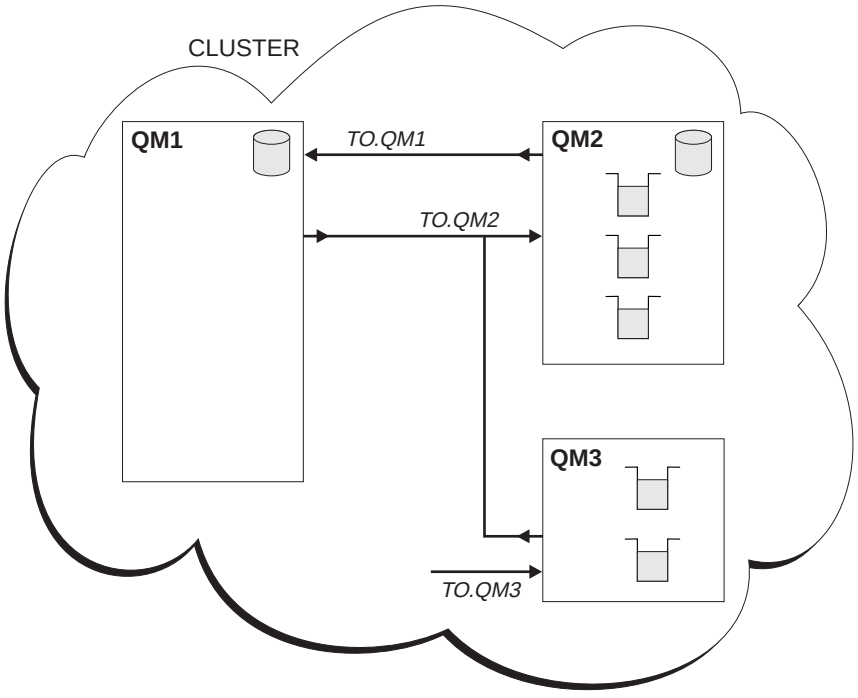

Figure 3. A cluster of queue managers

As you can see, the amount of system administration is greatly reduced, even with just a small cluster, containing two or three queue managers.

Distributed queuing

In addition to the reduction in system administration, clusters enable you to increase the availability of your queues and to distribute workload between your queue managers. Many queue managers can have definitions for the same queue. Any one of these queue managers can receive the message and so the workload can be shared between the queue managers.

MQSeries clusters are available on Version 5.1 of MQSeries for AIX, AS/400, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, and MQSeries for OS/390 Version 2.1. On Windows NT, you should take care when using clustering in an environment where IP addresses change on an unpredictable basis, such as on machines where Dynamic Host Configuration Protocol (DHCP) is being used. Specify the host name rather than the IP address in your channel definitions, otherwise you must update the IP address each time DHCP issues a new one.

How can I use MQSeries clusters?

You can set up a cluster, or convert an existing network of queue managers into a cluster, using a small number of definitions. Typically, a cluster contains queue managers that are logically related in some way and need to share some data or applications. For example, you might group all the departments in your company so that they all feed into the PAYROLL application, or you might group the branches of your department store so that they all feed into the SALES and PURCHASES applications. Once a cluster has been set up, the queue managers within it can communicate with each other without the need for any channel or remote queue definitions.

For more information about MQSeries clusters, see the *MQSeries Queue Manager Clusters* manual.

Where to find more information

You can find more information in the *MQSeries Intercommunication* manual and the *MQSeries Queue Manager Clusters* manual.

Also, you might find it useful to read the following redbooks published by the IBM International Technical Support Organization:

- *Examples of Using MQSeries on S/390®, RISC System/6000®, AS/400, and PS/2*, GG24-4326
- *Multiplatform APPC Configuration Guide*, GG24-4485

Request these books through your IBM representative.

Chapter 3. Introduction to MQSeries security

This chapter gives an overview of the security facilities that are provided by MQSeries products. The ways of using these facilities as part of the MQSeries framework are described under “Security enabling interface (SEI)” on page 58.

This chapter contains basic information about:
- “MQSeries security functions” on page 22
- “Where to find more information” on page 25

To understand the security functions provided by MQSeries, you need to understand the logical positioning of MQSeries relative to other components within a system and to understand the security services that are under discussion. The relative positioning of MQSeries is illustrated in Figure 4, which is an extract from the Open Blueprint®, where MQSeries is represented by the Messaging and Queuing component.

Figure 4 shows that security services are separated from other components such as application services, resource managers, and communications. This separation means that security services need only be provided by one component in the node and other components simply call the appropriate services when needed.

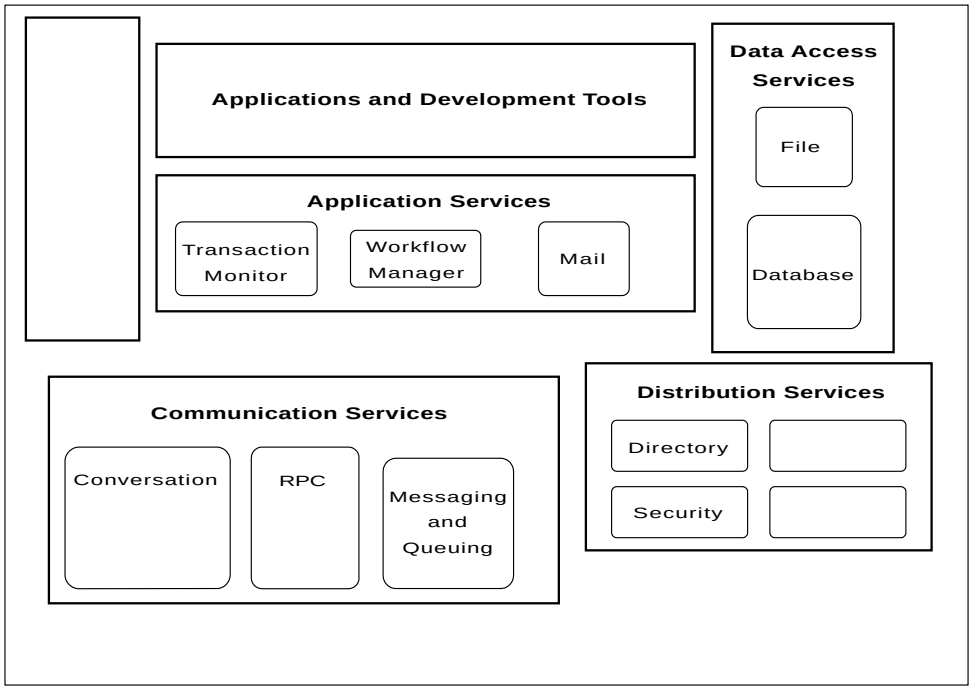

Figure 4. MQSeries and the Open Blueprint

The security services that can be provided by the (separate) security component are:

- Identification and authentication (I&A)

  This service forms the basis of many of the other services and involves the provision of a user identifier (user ID or principal) and the verification that the identifier is valid (that is, it represents the actual user and is not some intruder impersonating a valid user).
- Authorization

  This is access control and relies upon the availability of some user identifier to compare against access control lists (ACLs).

  Note that the authorization service is only useful if it is used; an intruder might attempt to bypass (and neutralize) the authorization service.
- Data confidentiality, or encryption
- Data integrity

  Even though data might be visible (that is, not encrypted), the data integrity service ensures that data is not altered.
- Non-repudiation

  This is the provision of some form of token (such as a digital signature) which guarantees that a particular piece of data originated from a particular user.

Note that although the Security component is positioned as one of the distribution services, it also provides services within the node (particularly authorization but all of the other services as well).

The way in which these security services are used by the various aspects of MQSeries is explained in the following sections.

MQSeries security functions

The following sections apply to all MQSeries products.

MQSeries applications

Before an application connects to a queue manager, it will have undergone some form of I&A procedure. This might be the provision of a user ID and password or might be some more elaborate process (such as smart card identification). Alternatively, it might be that there is no requirement for the I&A procedure and, thus, no user identifier associated with the application (for example, the building that houses the system might be physically secure and user identifiers might not be considered necessary).

The consequence of this is that each application that issues MQI calls has an associated user identifier (which might be null) that is used to authorize the use of certain MQI functions (primarily **MQCONN** and **MQOPEN**) and options (such as PUT and GET) against particular objects (usually queues). This means that any application user identifier trying to access MQSeries resources must be suitably authorized.

Because the I&A procedure takes place before the application connects to the queue manager, it is the responsibility of components other than MQSeries to provide the I&A service. MQSeries is responsible only for capturing the user identifier for use in providing other security services, such as authorization. (The user identifier is captured when the application connects to the queue manager.)

OS/2 Warp is the exception to this scope of responsibility, where MQSeries provides a service to provide a user identifier for applications when they connect to the queue manager. This is called the user identifier service and is documented in the *MQSeries Programmable System Management* manual. The user identifier service is part of the MQSeries framework; see "Security enabling interface (SEI)" on page 58 for further details. This service is provided because there are no security functions on this platform.

Similarly, it is often the responsibility of some other component to provide the authorization service, with MQSeries having responsibility for calling that service. This works satisfactorily for systems such as OS/390, where there is a standardized interface (for example, SAF) to the authorization service. For the OS/2 Warp, Windows NT, and UNIX platforms, however, there is no standard authorization service or interface provided and so MQSeries provides its own interface. This is called the authorization service and is documented in the *MQSeries Programmable System Management* manual. The authorization service is part of the MQSeries framework; see "Security enabling interface (SEI)" on page 58 for further details.

The Object Authority Manager

For MQSeries for AS/400, MQSeries for Compaq (DIGITAL) OpenVMS, MQSeries for Tandem NSK, MQSeries on UNIX systems, and MQSeries for Windows NT, there is an additional component written to the SEI that provides an authorization service. This is called the object authority manager (OAM) and it restricts access to MQSeries objects based upon the access control lists (ACLs) that it manages.

The OAM is documented in the:

- *MQSeries for AS/400 V5.1 System Administration* manual
- *MQSeries for Digital OpenVMS System Management Guide*
- *MQSeries for Digital UNIX System Management Guide*
- *MQSeries for Tandem NonStop Kernel System Management Guide*
- *MQSeries for VSE/ESA System Management Guide*
- *MQSeries System Administration* manual

The object authority manager is part of the MQSeries framework; see "Security enabling interface (SEI)" on page 58 for further details.

MQSeries messages

The basic function of MQSeries is to pass messages between applications. The message header (the message descriptor, MQMD) contains a field (named *UserIdentifier*) where a user identifier can be placed, allowing the application that gets the message to know from which user the message originated. The user identifier can be placed in the MQMD in one of three ways:

- The user identifier from a previous message (that is, one for which an **MQGET** has been performed) can be passed to a subsequent message. This is known as passing the security context.
- A suitably authorized (that is, trusted) application can place *any* user identifier in the field.
- If neither of the above is used, MQSeries automatically places in this field the user identifier of the application that did the **MQPUT**.

Therefore, this aspect of MQSeries operation provides an identification service (associating a user identifier with the message); it does *not* provide an authentication service. It is currently the responsibility of MQSeries applications both to provide any required authentication token (on the **MQPUT** side) and to verify that token (on the **MQGET** side).

Note also that, when an application does an **MQGET** for a message, there is no attempt to reset the application's user identifier to that contained in the message header. This function is called context management and is not supported by MQSeries.

Point-to-point security

In addition to providing services as a local resource manager, a queue manager also provides distributed queuing, enabling messages to be distributed around a network of queue managers. This distributed messaging function is provided by means of MQSeries channels. Each channel is composed of a pair of message channel agent programs (MCAs), which provide the protocol for assured, once-only message delivery using an underlying transport mechanism to exchange messages.

When the two MCAs establish communication, it might be necessary for each to verify the identity of the other. This would be the case if one queue manager did not trust the connection or the identity of the partner queue manager (for example, if they were owned by separate enterprises). This verification can be accomplished in one of the following ways:

1. Some transport mechanisms (in particular APPC) provide security features such as session authentication. Note that this provides verification of the partner system (the partner logical unit), rather than the partner application (the MCA) but this might satisfy the security requirements of the queue manager.
2. The MCAs each provide a security exit point which can be used to call user-written security exits for the exchange of user identifiers and associated authentication tokens (password, ticket, and so on). This allows each MCA to verify the identity of its partner.

   Using the MCA security exit allows the channel to be independent of the underlying transport mechanism and to provide a consistent service across many transports. This is especially important when providing a service (like security) that is available only on a limited set of transports.

   This aspect of MQSeries operation provides support for the I&A service. If required, the security exit can use the central security services to provide authentication tokens but this is not a requirement. See "Security enabling interface (SEI)" on page 58 for more information about MCA exits.
3. On AIX, HP-UX, OS/2 Warp, Sun Solaris, Windows NT, and Windows 95, MQSeries provides exits relating to DCE security. Source code is also provided to help you understand the program, and to assist you in creating your own.

Note that it is possible to use any or all of the above mechanisms for point-to-point security, enabling an MCA to be assured that it is exchanging messages with the correct queue manager.

End-to-end security

End-to-end security refers to security services that can be provided when a message is PUT by an application and to the corresponding services that are available when the target application performs a GET. The relevant services are identification and authentication (possibly across the network), data confidentiality, data integrity, and non-repudiation.

MQSeries is not responsible for the provision of these services but it is responsible for providing appropriate interfaces to call these services.

Today, the only aspect of end-to-end security that is (directly) supported by MQSeries is Identification, where a user identifier can be placed in a message header. MQSeries does not provide (direct) support for any of the other services. However, these services can be implemented in either MQSeries application programs or the MCA message exit, which is a customer exit invoked each time a message is passed between two queue managers.

Where to find more information

For information about MQSeries security on your platform, see the following chapters:

- “Chapter 11. Security planning for MQSeries for AS/400” on page 69
- “Chapter 16. Security planning for MQSeries for Compaq (DIGITAL) OpenVMS” on page 89
- “Chapter 21. Security planning for MQSeries for OS/2 Warp” on page 109
- “Chapter 27. Security planning for MQSeries for OS/390” on page 133
- “Chapter 34. Security planning for MQSeries for Tandem NSK” on page 161
- “Chapter 38. Security planning for MQSeries for UNIX systems” on page 177
- “Chapter 41. Introduction to MQSeries for VSE/ESA” on page 187
- “Chapter 44. Security planning for MQSeries for Windows NT” on page 203

For information about the security enabling interface, see “Chapter 8. Introduction to the MQSeries Framework” on page 55.

Chapter 4. Introduction to MQSeries recovery concepts

MQSeries products provide facilities for applications to be able to keep related sets of data consistent when changes are made to them. This chapter describes the background concepts of recovery and restart that you will need to understand before going on to the later chapters in this book.

This chapter gives basic information about:

How changes are made to data

You need to understand how MQSeries queue managers interact with other programs to keep all the data consistent. This section discusses *units of recovery*.

Units of recovery

In a single queue manager, a *unit of recovery* is a piece of work that changes data from one point of consistency to another. A *point of consistency* (also called a *syncpoint* or *commit point*) is a moment at which all the recoverable data that an application program accesses is consistent. This means that a unit of recovery begins with an attempt to change data and ends at a point of consistency. An example of a unit of recovery within an application program is shown in Figure 5.

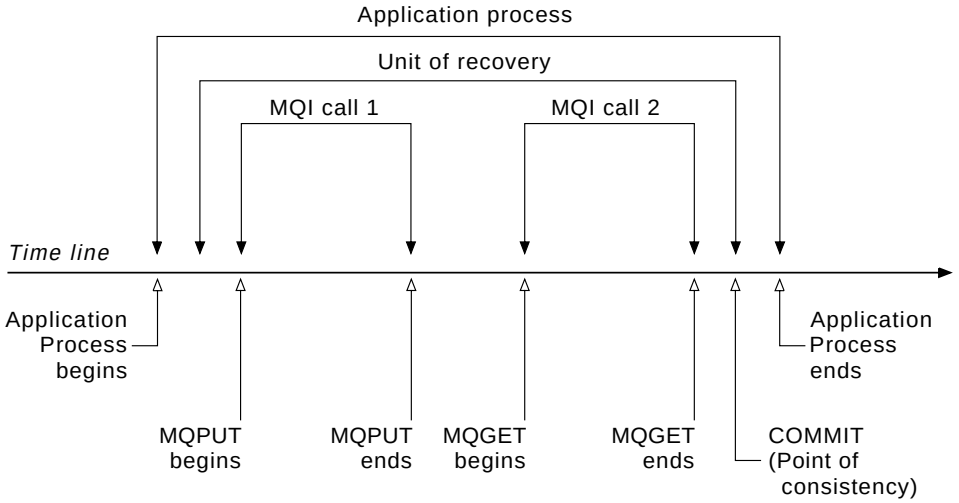

Figure 5. A unit of recovery within an application

In this example, the application process makes changes to queues at *MQI call 1* and *MQI call 2*. The application process can include a number of units of recovery or just one, but any complete unit of recovery ends with a commit point.

For example, a bank transaction might transfer funds from account A to account B. First, a debit program subtracts the amount from account A. Next, account B must be credited with the amount. After subtracting the amount from account A, the two accounts are inconsistent and the queue manager cannot commit. The next step in the process is for the debit program to put a message on a queue to the credit program, giving information about the amount of money and the account to be credited. When the credit program gets the message from the queue, it performs the credit to account B and informs the debit program that it has completed the task. The data is now consistent. The application can announce a point of consistency and make the changes visible to other applications.

A point of consistency should be created by an application before it terminates. It can do this by issuing a commit call. Normal termination of an application on some platforms automatically causes a point of consistency, but you should verify that this will occur on your platform.

If an error occurs within a unit of recovery, you can use the queue manager to remove changes to data, returning the data to its state at the start of the unit of recovery; that is, the queue manager undoes the work.

Some systems have syncpoint facilities. In these cases it is the system function that causes a syncpoint to be created for both that system's data, and the data that is being managed by MQSeries queue managers, simultaneously.

MQSeries products can operate with other products (such as transaction processors) to provide coordination between MQSeries and non-MQSeries resources. Table 2 lists those products that can work in this way with the latest MQSeries server products.

Table 2. MQSeries and cooperating products

MQSeries product	**Cooperating products**
MQSeries for AIX	BEA Tuxedo, TXSeries™ for AIX
MQSeries for AS/400	CICS for AS/400
MQSeries for AT&T GIS UNIX	BEA Tuxedo
MQSeries for Compaq (DIGITAL) OpenVMS	—
MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX)	—
MQSeries for HP-UX	BEA Tuxedo, TXSeries for HP-UX
MQSeries for OS/2 Warp	CICS Transaction Server for OS/2
MQSeries for OS/390	CICS for MVS/ESA™, IMS/ESA®, OS/390 RRS, CICS Transaction Server for OS/390
MQSeries for SINIX and DC/OSx	CICS for SINIX, Encina, Novell Tuxedo
MQSeries for Sun Solaris	BEA Tuxedo, TXSeries for Sun Solaris
MQSeries for Tandem NonStop Kernel	TM/MP (TMF)
MQSeries for VSE/ESA	CICS/VSE®
MQSeries for Windows NT	BEA Tuxedo, TXSeries for Windows NT

How consistency is maintained

If the data being managed by a queue manager is to be consistent with the data in other subsystems, any data changed in one must be matched by a change in the others. Before one system commits the changed data, it must know that the other system, or systems, can make the corresponding changes. So, the systems must communicate.

During a *two-phase commit* one subsystem coordinates the process. That subsystem is called the *coordinator*; the others are the *participants*.

- On AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, MQSeries can be either the coordinator or a participant.
- On OS/400, OS/400 is always the coordinator in interactions with MQSeries, and MQSeries is always a participant.
- On other platforms, CICS, IMS™, or RRS is always the coordinator in interactions with MQSeries, and MQSeries is always a participant.

The coordinator first checks that all participants are ready to perform the commit; that is phase one. When the coordinator receives positive replies from the participants that they are ready to commit, it issues the commit request, this being phase 2. Together with the coordinator, all participants now commit the changed data. All data is once again consistent.

During a *one-phase commit* there is no coordinator as such in the interactions. When the commit call is issued, the response to the call is that:

- The commit was successful.
- The call was partially complete.
- The call failed.

It is up to the application issuing the call to decide what further action, if any, needs to be taken.

When the queue manager is restarted after an abnormal termination, it must determine whether to commit or to back out units of recovery that were active at the time of the error. For certain units of recovery, the queue manager has enough information to make the decision. For others, it does not, and must get information from the coordinator when the connection is reestablished.

Syncpointing cannot occur directly across remote links. Instead, use is made of the assured message delivery feature of the MQSeries products. A commit request is sent in a message to a remote system by an application with the assurance that the message will be delivered. The application designers must ensure that they create, and respond to, commit requests. They must also allow for situations where the commit cannot be accepted, and respond accordingly to the commit requester.

An example of the problems of remote commits is when the remote system is not available. The messages to it will remain on the transmission queues until they can be delivered. When delivery eventually occurs, any desired commitments can then be made. In this situation, we are dealing with time independent applications. It is for the application designers to consider how, and when, they require the data shared between their remote systems to be consistent.

MQSeries as a transaction manager

On some platforms, MQSeries can act as a transaction manager and will coordinate updates made by external resource managers within MQSeries units of work. These external resource managers must comply to the X/Open XA interface.

Table 3 shows the MQSeries platforms that can act as a transaction manager, and the XA-compliant database managers they support. Figure 6 through Figure 8 on page 31 show how you must configure your queue manager and database servers to enable this.

Table 3. XA-compliant database managers supported by MQSeries

Platform	DB2®	Oracle	Sybase
AIX	✓	✓	✓
HP-UX	✓	✓	
OS/2 Warp	✓		
OS/400	✓		
Sun Solaris	✓	✓	✓
Windows NT	✓		✓

The following figures describe the possible configurations for your queue manager and database. The configuration shown in Figure 6 is allowed:

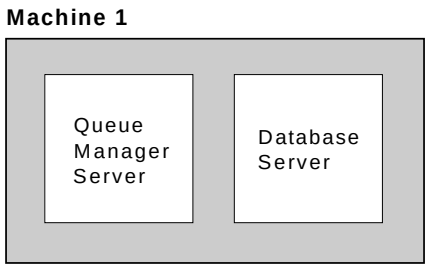

Figure 6. Queue manager and database server configuration 1. Queue manager server and database server on the same machine.

The configuration shown in Figure 7 is also allowed provided that you use appropriate database connection features (for example, CAE/DDCS for DB2):

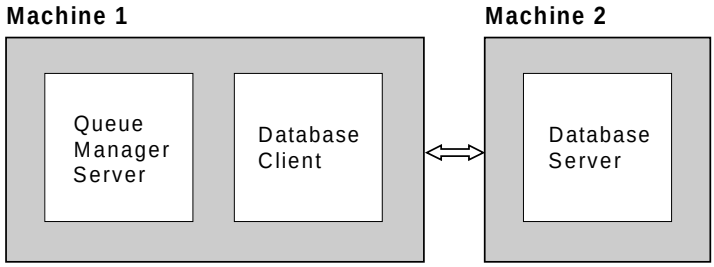

Figure 7. Queue manager and database server configuration 2. Queue manager server and database client on the same machine.

The configuration shown in Figure 8 is not allowed because coordination can be provided only by the MQSeries server, not the client:

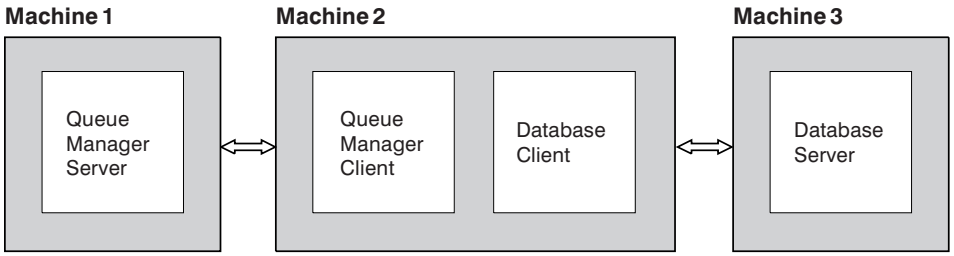

Figure 8. Queue manager and database server configuration 3. Queue manager server and database server on different machine.

Where to find more information

For information about MQSeries recovery on your platform, see the following chapters:

- "Chapter 10. Backup and recovery planning for MQSeries for AS/400" on page 67
- "Chapter 15. Backup and recovery planning for MQSeries for Compaq (DIGITAL) OpenVMS" on page 85
- "Chapter 20. Backup and recovery planning for MQSeries for OS/2 Warp" on page 105
- "Chapter 26. Backup and recovery planning for MQSeries for OS/390" on page 127
- "Chapter 33. Backup and recovery planning for MQSeries for Tandem NSK" on page 159
- "Chapter 37. Backup and recovery planning for MQSeries for UNIX systems" on page 173
- "Chapter 41. Introduction to MQSeries for VSE/ESA" on page 187
- "Chapter 43. Backup and recovery planning for MQSeries for Windows NT" on page 199

Chapter 5. Introduction to MQSeries administration

This chapter gives a summary of the administration facilities that are provided by the MQSeries products.

Administration of an application that uses MQSeries products is performed by a system administrator, system programmer, or computer operator who has the appropriate authority. It is the administrator's responsibility to monitor the MQSeries products, and the resources that they are using, and make any changes that might be necessary to keep the applications running without problems.

This chapter has the following sections:

Some of the facilities described in this chapter are not supported by all MQSeries products.

Introducing commands

MQSeries products provide a set of commands, command interfaces, and utilities that provide messaging and queuing administration functions, including, for example:

Queue administration	Creating and deleting queues
Channel administration	Stopping or starting channels
Security administration	Changing the access authority to queues

These functions are provided by *commands* in MQSeries, which are processed, by the queue manager, in the following ways:

- **Entered by the system administrator**

  A command is entered by a system administrator and processed immediately by the queue manager. A response is given to the administrator.

  The method used to enter the commands is platform dependent, and is the normal method that is used for entering commands on the platform. This could be a command console, a command line, a display panel utility, or some other technique.

- **Stored as a list of commands**

  The commands are stored as a list in a file. Later, the administrator can cause the queue manager to read the commands in the file, process them, and generate responses.

- **Stored as messages in a queue**

  An application program generates a command, and puts it to a queue manager *command queue*, using the MQPUT call. The queue manager gets the message off the queue, processes the command that is contained in the message, and generates a response.

Formats of command

The MQSeries products provide the following types of command:
- **MQSeries commands (MQSC)** - human-readable
- **Programmable Command Format (PCF) commands** - machine-readable

Both sets of commands perform similar functions, but are intended for use in different situations.

In addition to the MQSC and PCF commands, the MQSeries products provide *control commands*. These are the commands used by the system administrator to pass control information to the queue manager, and are designed to be compatible with the other administrative commands used on a particular platform. Because of the differences in the way that administrative commands are processed by the various operating systems in which the MQSeries products operate, the technique used to enter control commands varies with the particular MQSeries product.

MQSC commands

MQSeries commands (MQSC) provide a uniform method of issuing commands in a human-readable form across MQSeries platforms. They are intended for use in those situations where people have to be able to read the commands, for example, whenever commands are entered through a command line. The responses to these commands are also human-readable, however the content and format depends on the platform in which they are used.

The commands can be issued from the following sources:
- Entered through the command line
- Stored in a file, and processed by the queue manager
- Stored within messages on a command queue

MQSC commands are described in the *MQSeries Command Reference* manual. They are not available on MQSeries for VSE/ESA.

Programmable Command Format commands

MQSeries Programmable Command Format (PCF) commands are intended for use by application programs that provide facilities for the administration of queue managers and the resources that they are using (for example, queues and channels), from a single point within the network. The format of these commands, and their responses, is independent of the environment in which they are used.

PCFs define commands and reply messages that can be exchanged between a program and any queue manager in a network.

You can use PCF commands in a systems-management application program for administration of MQSeries objects: queue managers, process definitions, queues, and channels. The application can operate from a single point in the network to communicate command and reply information with any queue manager, local or remote, via the local queue manager.

It is possible, by using the PCF **escape** command, to include the MQSC command as character strings within a PCF command.

PCF commands are described in the *MQSeries Programmable System Management* manual. They are not available on MQSeries for OS/390 or MQSeries for VSE/ESA.

Command queues

Some MQSeries products have a command queue that accepts administration messages containing commands, called *command messages*. The queue manager has a *command server* that processes the command messages from the command queue.

Command messages can be placed on the queue by any local or remote application program. The command messages are processed by the command server, and a response is returned to the application program using the name of the reply-to queue that is contained within the original command message. PCF commands and reply messages are sent and received using the MQPUT and MQGET calls defined in the Message Queue Interface (MQI).

There are two types of command queue, each with system-defined queue names.

1. SYSTEM.ADMIN.COMMAND.QUEUE

   This queue accepts PCF commands only, including the **escape** command. This type of queue is supported by the following:
   - MQSeries for AS/400
   - MQSeries for Compaq (DIGITAL) OpenVMS
   - MQSeries for OS/2 Warp
   - MQSeries for Tandem NSK
   - MQSeries on UNIX systems
   - MQSeries for Windows NT
   - MQSeries for Windows (Version 2.1)
2. SYSTEM.COMMAND.INPUT

   This queue accepts MQSC commands only. This type of queue is supported by MQSeries for OS/390 only.

The MQSeries products that are not mentioned above do not have command queues.

Administration application programs

MQSeries products provide application programs, called *administration utilities*, that allow the system administrator on one processor to control the MQSeries product that is running on that processor. Some of these utilities also allow the administrator to control MQSeries products that are running on other processors, so-called *remote administration*.

The MQSeries Explorer, supplied with MQSeries for Windows NT, provides a graphical user interface for administering your MQSeries network. The MQSeries Explorer cannot be used to control queue managers on OS/390 or VSE/ESA.

In those situations where no administration utility is available, you can write your own application program to provide those functions. This application can manage any queue manager (local or remote) that has a command queue, by putting command messages to that queue using the MQPUT call. When a command is processed, a response message is returned to the reply-to queue that your application program specified in the original command message.

MQSeries product administration facilities

More information about the control commands and administration facilities provided by the MQSeries server products is given in later sections of this manual, as follows:

- **MQSeries for AS/400** – see "Chapter 12. Administration of MQSeries for AS/400" on page 73.
- **MQSeries for Compaq (DIGITAL) OpenVMS** – see "Chapter 17. Administration of MQSeries for Compaq (DIGITAL) OpenVMS" on page 93.
- **MQSeries for OS/2 Warp** – see "Chapter 22. Administration of MQSeries for OS/2 Warp" on page 111.
- **MQSeries for OS/390** – see "Chapter 28. Administration of MQSeries for OS/390" on page 137.
- **MQSeries for Tandem NonStop Kernel** – see "Chapter 35. Administration of MQSeries for Tandem NSK" on page 163.
- **MQSeries on UNIX systems** – see "Chapter 39. Administration of MQSeries on UNIX systems" on page 181.
- **MQSeries for VSE/ESA** – see "Administration of MQSeries for VSE/ESA" on page 189.
- **MQSeries for Windows NT** – see "Chapter 45. Administration of MQSeries for Windows NT" on page 205.
- **MQSeries for Windows** – see "Chapter 47. Introduction to MQSeries for Windows" on page 213.

A tabular summary of the commands supported by these products is given in "Command summary" on page 37.

The MQSeries Administration Interface

The MQSeries Administration Interface (MQAI) is a programming interface to MQSeries. It performs administration tasks on an MQSeries queue manager using *data bags*. Data bags allow you to handle properties (or parameters) of objects in a way that is simpler than using the Programmable Command Format (PCF) administration interface.

The MQAI is available on MQSeries for AIX, AS/400, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT. It can be used for the following tasks:

- To simplify the use of PCF messages. The MQAI is an easy way to administer MQSeries; you do not have to write your own PCF messages and this avoids the problems associated with complex data structures.
- To implement self-administering applications and administration tools. For example, the Active Directory Service Interfaces provided on Windows NT uses the MQAI.
- To handle error conditions more easily. It is difficult to get return codes back from the MQSeries commands (MQSC), but the MQAI makes it easier for the program to handle error conditions.

For more information about using the MQAI, see the *MQSeries Administration Interface Programming Guide and Reference* manual.

Command summary

The following tables show how the various command formats in MQSeries relate to each other. The command formats available are:

- Programmable command format (PCF) commands

  PCF commands are not supported on OS/390 or VSE/ESA.
- MQSeries commands (MQSC)

  MQSC are not supported on VSE/ESA.
- CL commands

  CL commands are supported on OS/400 only.
- Control commands

  Control commands are supported on Compaq (DIGITAL) OpenVMS, OS/2 Warp, Tandem NonStop Kernel, UNIX systems, and Windows NT.

In the following tables, an empty cell indicates that there is no equivalent command in the specified format.

Table 4. Commands for queue manager administration

Operation	PCF	MQSC	OS/400 CL	Control command
Change Queue Manager attributes	Change Queue Manager	ALTER QMGR or DEFINE MAXSMSGS (1)	CHGMQM	
Connect a Queue Manager			CCTMQM	
Create a Queue Manager			CRTMQM	crtmqm
Delete a Queue Manager			DLTMQM	dltmqm
Disconnect from a Queue Manager			DSCMQM	
Display Queue Manager attributes	Inquire Queue Manager	DISPLAY QMGR or DISPLAY MAXSMSGS (1)	DSPMQM	
Install MQSeries for Tandem NonStop Kernel				instmqm (2)
Ping a Queue Manager	Ping Queue Manager	PING QMGR (3)		
Start a Queue Manager		START QMGR (1)	STRMQM	strmqm
Stop a Queue Manager		STOP QMGR (1)	ENDMQM	endmqm
Upgrade MQSeries for Tandem NonStop Kernel				upgmqm (2)
Work with a Queue Manager			WRKMQM	

Notes:

1. Applies on OS/390 only
2. Applies on Tandem NSK only
3. Does not apply on OS/390

Administration overview

Table 5. Commands for queue administration

Operation	PCF	MQSC	OS/400 CL	Control command
Alter queue file attributes				altmqfls (1)
Change queue attributes	Change Queue	ALTER QALIAS ALTER QLOCAL ALTER QMODEL ALTER QREMOTE	CHGMQMQ	
Clear a queue	Clear Queue	CLEAR QLOCAL (2) The following sequence: DELETE QLOCAL(x), DEFINE QLOCAL(x) or the following sequence: DEFINE QLOCAL(y) LIKE(x), DELETE QLOCAL(x), DEFINE QLOCAL(x) LIKE(y), DELETE QLOCAL(y)	CLRMQMQ	
Copy a queue definition	Copy Queue	DEFINE QALIAS(x) LIKE(y) DEFINE QLOCAL(x) LIKE(y) DEFINE QMODEL(x) LIKE(y) DEFINE QREMOTE(x) LIKE(y)	CPYMQMQ	
Create a queue	Create Queue	DEFINE QALIAS DEFINE QLOCAL DEFINE QMODEL DEFINE QREMOTE	CRTMQMQ	
Delete a queue	Delete Queue	DELETE QALIAS DELETE QLOCAL DELETE QMODEL DELETE QREMOTE	DLTMQMQ	
Display queue attributes	Inquire Queue	DISPLAY QUEUE DISPLAY QALIAS (3) DISPLAY QCLUSTER (3) DISPLAY QLOCAL (3) DISPLAY QMODEL (3) DISPLAY QREMOTE (3)	DSPMQMQ	
Display queue names	Inquire Queue Names	DISPLAY QUEUE	WRKMQMQ	
Reset queue statistics	Reset Queue Statistics			
Work with a queue			WRKMQMQ	
Work with messages			WRKMQMMSG	

Notes:

1. Applies on Tandem NSK only
2. Does not apply on OS/390
3. Applies on AIX, HP-UX, OS/2 Warp, OS/390, OS/400, Sun Solaris, and Windows NT only

Table 6. Commands for process definition administration. (See note 1.)

Operation	PCF	MQSC	OS/400 CL
Change process attributes	Change Process	ALTER PROCESS	CHGMQMPRC
Copy a process	Copy Process	DEFINE PROCESS(x) LIKE(y)	CPYMQMPRC
Create a process	Create Process	DEFINE PROCESS	CRTMQMPRC
Delete a process	Delete Process	DELETE PROCESS	DLTMQMPRC
Display process attributes	Inquire Process	DISPLAY PROCESS	DSPMQMPRC
Display process names	Inquire Process Names	DISPLAY PROCESS	WRKMQMPRC
Work with a process			WRKMQMPRC

Note:

1. There are no control-command equivalents of these commands.

Table 7. Commands for namelist administration. (See notes 1 and 2.)

Operation	PCF (3)	MQSC	OS/400 CL
Change a namelist	Change Namelist	ALTER NAMELIST	CHGMQMNL
Copy a namelist	Copy Namelist	DEFINE NAMELIST(x) LIKE(y)	CPYMQMNL
Define a namelist	Create Namelist	DEFINE NAMELIST	CRTMQMNL
Delete a namelist	Delete Namelist	DELETE NAMELIST	DLTMQMNL
Display a namelist	Inquire Namelist	DISPLAY NAMELIST	DSPMQMNL
Display namelist names	Inquire Namelist Names	DISPLAY NAMELIST	WRKMQMNL
Work with a namelist			WRKMQMNL

Notes:

1. Namelist functions are supported on AIX, HP-UX, OS/2 Warp, OS/390, OS/400, Sun Solaris, and Windows NT only.
2. There are no control-command equivalents of these commands.
3. Does not apply on OS/390.

Administration overview

Table 8. Commands for channel administration

Operation	PCF	MQSC (1)	OS/400 CL	Control command
Change channel attributes	Change Channel	ALTER CHANNEL	CHGMQMCHL	
Convert client channel definitions				cnvclchl (2)
Copy channel attributes	Copy Channel	DEFINE CHANNEL (x) LIKE (y)	CPYMQMCHL	
Create a channel	Create Channel	DEFINE CHANNEL	CRTMQMCHL	
Delete a channel	Delete Channel	DELETE CHANNEL	DLTMQMCHL	
Display a channel	Inquire Channel	DISPLAY CHANNEL	DSPMQMCHL	
Display channel names	Inquire Channel Names	DISPLAY CHANNEL	WRKMQMCHL	
Display channel status	Inquire Channel Status	DISPLAY CHSTATUS		
Display distributed queuing		DISPLAY DQM (3)		
Ping a channel	Ping Channel	PING CHANNEL	PNGMQMCHL	
Reset a channel	Reset Channel	RESET CHANNEL	RSTMQMCHL	
Resolve a channel	Resolve Channel	RESOLVE CHANNEL	RSVMQMCHL	
Start a channel	Start Channel	START CHANNEL	STRMQMCHL	runmqchl
Start a channel initiator (4)	Start Channel Initiator	START CHINIT (5)	STRMQMCHLI	runmqchi
Start a channel listener (4)	Start Channel Listener (6)	START LISTENER (7)	STRMQMLSR	runmqlsr (8)
Stop a channel	Stop Channel	STOP CHANNEL	ENDMQMCHL	
Stop a channel initiator (4)		STOP CHINIT (3)		
Stop a channel listener (4)		STOP LISTENER (3)	ENDMQMLSR	endmqlsr (9)
Work with channel status			WRKMQMCHST	

Notes:

1. Does not apply on OS/390 if you are using CICS for distributed queuing.
2. Applies on Tandem NSK only.
3. Applies on OS/390 only.
4. In MQSeries for Tandem NonStop Kernel, the preferred method of starting and stopping TCP/IP listeners and channel initiators is using PATHWAY (TS/MP), as described in the *MQSeries for Tandem NonStop Kernel System Management Guide*.
5. Does not apply on Tandem NSK.
6. Applies on OS/2 Warp, OS/400, and Windows NT only.
7. Applies on AIX, HP-UX, OS/2 Warp, OS/390, OS/400, Sun Solaris, and Windows NT only.
8. Does not apply on AT&T GIS UNIX, DIGITAL UNIX, or SINIX and DC/OSx.
9. Applies on AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT only.

Table 9. Commands for cluster administration. (See notes 1 and 2.)

Operation	PCF (3)	MQSC	OS/400 CL
Display cluster queue manager	Inquire Cluster Queue Manager	DISPLAY CLUSQMGR	
Refresh cluster information	Refresh Cluster	REFRESH CLUSTER	RFRMQMCL
Reset cluster	Reset Cluster	RESET CLUSTER	RSTMQMCL
Resume cluster processing	Resume Queue Manager Cluster	RESUME QMGR	RSMMQMCLQM
Suspend cluster processing	Suspend Queue Manager Cluster	SUSPEND QMGR	SPDMQMCLQM
Work with a cluster			WRKMQMCL

Notes:

1. Queue Manager Cluster functions are supported on AIX, HP-UX, OS/2 Warp, OS/390, OS/400, Sun Solaris, and Windows NT only.
2. There are no control-command equivalents of these commands.
3. Does not apply on OS/390.

Table 10. Commands for security administration. (See note 1.)

Operation	MQSC (2)	OS/400 CL	Control command
Alter security options	ALTER SECURITY		
Display mapping of MQSeries principal to Tandem NSK user ID			dspmqusr (3)
Display object authority		DSPMQMAUT	dspmqaut
Display security settings	DISPLAY SECURITY		dspmqaut
Grant object authority		GRTMQMAUT	setmqaut
Map MQSeries principal to Tandem NSK user ID or group			altmqusr (3)
Refresh security	REFRESH SECURITY		
Revoke object authority		RVKMQMAUT	setmqaut
Set a reverification flag	RVERIFY SECURITY		

Notes:

1. There are no PCF equivalents of these commands.
2. Applies on OS/390 only.
3. Applies on Tandem NSK only.

Table 11. Miscellaneous commands. (See note 1.)

Operation	MQSC (2)	OS/400 CL	Control command
Alter a storage class	ALTER STGCLASS		
Alter trace parameters	ALTER TRACE		
Archive a log	ARCHIVE LOG		
Convert V1.5.1 messages to V2.2.0.1 format			cnvmsgs (3)
Convert V1.5.1 queue and channel definitions to V2.2.0.1 format			cnv1520 (3)
Create data conversion code		CVTMQMDTA	crtmqcvx
Define a buffer pool	DEFINE BUFFPOOL		
Define a page set	DEFINE PSID		
Define a storage class	DEFINE STGCLASS		
Delete a storage class	DELETE STGCLASS		

Table 11. Miscellaneous commands (continued). (See note 1.)

Operation	MQSC (2)	OS/400 CL	Control command
Display the command server	DISPLAY CMDSERV	DSPMQMCSVR	dspmqcsv (4)
Display MQSeries files			dspmqfls
Display MQSeries formatted trace output			dspmqtrc (5)
Display MQSeries transactions		WRKMQMTRN	dspmqtrn (6)
Display an object name		DSPMQMOBJN	
Display page set information	DISPLAY USAGE		
Display storage class information	DISPLAY STGCLASS		
Display a thread	DISPLAY THREAD		
Display trace activity	DISPLAY TRACE		
Dump contents of MQSeries log			dmpmqlog (7)
Perform RDF housekeeping			cleanrdf (3)
Record an object image		RCDMQMIMG	rcdmqimg (8)
Recover a bootstrap data set	RECOVER BSDS		
Recreate an object		RCRMQMOBJ	rcrmqobj (8)
Reset an IMS transaction pipe	RESET TPIPE		
Resolve in-doubt threads	RESOLVE INDOUBT		
Resolve MQSeries transactions		RSVMQMTRN	rsvmqtrn (6)
Run dead-letter queue handler		STRMQMDLQ	runmqdlq
Run MQSeries commands		STRMQMMQSC	runmqsc
Start client trigger monitor			runmqtmc (9)
Start the command server	START CMDSERV	STRMQMCSVR	strmqcsv (4)
Start a trace	START TRACE	TRCMQM	strmqtrc (10)
Start trigger monitor		STRMQMTRM	runmqtrm
Stop the command server	STOP CMDSERV	ENDMQMCSVR	endmqcsv (4)
Stop a trace	STOP TRACE	TRCMQM	endmqtrc (10)

Notes:

1. There are no PCF equivalents of these commands.
2. Applies on OS/390 only.
3. Applies on Tandem NSK only.
4. In MQSeries for Tandem NonStop Kernel, as an alternative to the control commands **dspmqcsv**,**strmqcsv**, and**endmqcsv**, you can use PATHCOM commands to stop, start, and monitor the command server.
5. Does not apply on AIX, OS/2 Warp, or Windows NT.
6. Does not apply on DIGITAL UNIX or Tandem NSK.
7. Applies on AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT.
8. Does not apply on Tandem NSK.
9. Applies on AIX, Compaq (DIGITAL) OpenVMS, and OS/2 Warp clients only.
10. Does not apply on AIX.

Where to find more information

For information about MQSeries administration on your platform, see the following chapters:

- "Chapter 12. Administration of MQSeries for AS/400" on page 73
- "Chapter 17. Administration of MQSeries for Compaq (DIGITAL) OpenVMS" on page 93
- "Chapter 22. Administration of MQSeries for OS/2 Warp" on page 111
- "Chapter 28. Administration of MQSeries for OS/390" on page 137
- "Chapter 47. Introduction to MQSeries for Windows" on page 213
- "Chapter 35. Administration of MQSeries for Tandem NSK" on page 163
- "Chapter 39. Administration of MQSeries on UNIX systems" on page 181
- "Chapter 41. Introduction to MQSeries for VSE/ESA" on page 187
- "Chapter 45. Administration of MQSeries for Windows NT" on page 205

Chapter 6. Introduction to MQSeries instrumentation events

You can use the MQSeries instrumentation events to monitor the operation of queue managers. This chapter tells you what these events are, and describes how they can be used for system measurement and system management purposes.

The chapter contains these sections:
- "Monitoring queue managers"
- "What is an instrumentation event?"
- "Format of event messages" on page 48
- "Where to find more information" on page 48

Instrumentation events are supported by the following:
- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/2 Warp
- MQSeries for OS/390
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT
- MQSeries for Windows Version 2.1

Monitoring queue managers

An *instrumentation event* causes a special message, called an *event message*, to be generated by a queue manager whenever a set of predefined conditions occurs within the execution of your application.

You can write a system-monitoring application that collects event messages from many queue managers on different platforms, analyses them, and presents them to an administrator in summary form. This would allow you to monitor all the MQSeries products in your system from a single node.

Instrumentation events also enable applications acting as agents for other administration networks, for example NetView®, to monitor events and create the appropriate alerts.

What is an instrumentation event?

In MQSeries an instrumentation event is a logical combination of conditions that is detected by a queue manager. The result of such an event is that the queue manager puts a special message, called an *event message*, on an event queue.

For example, the conditions giving rise to a *queue full* event are:
- Queue full events are enabled for a specified queue.

and
- An application issues an **MQPUT** call to put a message on that queue, but the call fails because the queue is full.

Instrumentation events

Other conditions can also give rise to instrumentation events. For example:

- A threshold for the number of messages on a queue is reached.
- A channel instance is started or stopped.
- An application attempts to open a queue specifying a user ID that is not authorized.

Some instrumentation events must be enabled before they can be generated; this is described in “Enabling and disabling events” on page 47.

What types of event are there?

MQSeries events can be categorized as follows:

Queue manager events
These events are related to the definitions of resources within queue managers. For example, when an application attempts to put a message to a queue that does not exist.

Performance events
These events are notifications that a threshold has been reached by a resource. For example, when a queue depth threshold has been reached or, following a get, when the queue was not serviced within a predefined time limit.

Channel events
These events are reported by channels as a result of conditions detected during their operation. For example, when a channel instance is stopped.

Event notification through event queues

When an event occurs, the queue manager puts an event message on the appropriate event queue. The event message contains information about the event that you can retrieve by writing a suitable MQSeries application program that:

- Gets the message from the queue
- Processes the message to extract the event data

Each category of event has its own event queue. All events in that category result in an event message being put onto the same queue.

Table 12. Event queue contents

This event queue...	Contains messages from...
SYSTEM.ADMIN.QMGR.EVENT	Queue manager events
SYSTEM.ADMIN.PERFM.EVENT	Performance events
SYSTEM.ADMIN.CHANNEL.EVENT	Channel events

You can define event queues as either local or remote queues. If you define all your event queues as remote queues on the same queue manager, you can centralize your monitoring activities.

You must not define event queues as transmission queues, or initiation queues, because event messages have formats that are incompatible with the formats of messages required for those queues.

Using triggered event queues

You can set up the event queues with triggers, so that when an event is generated, the event message being put onto the event queue starts a (user-written) monitoring application. This application can process the event messages and take appropriate action. For example, certain events might require that an operator be informed, while other events might start off an application that performs some administration tasks automatically.

When an event queue is unavailable

If an event occurs when the event queue is not available, the event message is lost. For example, if you do not define an event queue for a category of event, all event messages for that category will be lost. The event messages are *not*, for example, saved on the dead-letter queue. An event queue might be unavailable for many different reasons, for example:

- The queue has not been defined.
- The queue has been deleted.
- The queue is full.
- The queue has been put-inhibited.

The absence of an event queue does not prevent the event from occurring. For example, after a performance event, the queue manager changes the queue attributes and might, depending on the event, reset the queue statistics. This happens whether or not the event message is put on the performance event queue.

Enabling and disabling events

You can enable and disable events by specifying the appropriate values for queue manager or queue attributes, or both, depending on the type of event. Commands are provided with the appropriate MQSeries products to allow you to set these attributes.

Note: Attributes related to events for both queues and queue managers can be set and inquired upon by command only. They are not supported by the MQI functions **MQSET** and **MQINQ**.

How you enable and disable events depends on the category of the event:

- Queue manager events are enabled by setting attributes on the queue manager.
- Performance events as a whole must be enabled on the queue manager, otherwise no performance events can occur. Then, you enable the specific performance events by setting the appropriate queue attribute. You also have to specify the conditions that give rise to the event.
- Channel events do not require enabling, they occur automatically. Similarly, channel events cannot be disabled.

Format of event messages

Event messages contain information about the event and its origin. Typically, these messages are processed by a system-management application program that is tailored to meet the requirements of the site at which it runs.

As with all MQSeries messages, an event message has two parts: a message descriptor, and the message data.

Message descriptor

The descriptor of an event message is a standard descriptor, as defined in the *MQSeries Application Programming Reference* manual. Some of the values in the descriptor might be of particular interest to a system-monitoring application, for example, the date and time when the event message was put on the event queue.

Event message data

The message data specifies:

- That the message is an event message.
- The category of the event, that is, whether the event is a queue manager, performance, or channel event.
- A reason code specifying the cause of the event.
- Event data specific to the event. This includes the name of the queue manager and, where appropriate, the name of the queue.

Where to find more information

More information about how to use events and details of the various types of event message can be found in the *MQSeries Programmable System Management* manual.

Information about enabling events can be found in the *MQSeries Command Reference* manual.

Chapter 7. Introduction to MQSeries clients and servers

This chapter introduces the concept of MQSeries clients and servers and describes how they can be of benefit in your MQSeries installation. It gives you information about some of the items that you need to consider when you are planning your applications.

The chapter has the following sections:

What are MQSeries clients and servers?

An *MQSeries client* is a part of an MQSeries product that can be installed on a machine without installing the full queue manager. It accepts Message Queue Interface (MQI) calls from application programs, and passes MQI requests to an *MQSeries server* that is executing on another processor.

The *MQSeries server* is a full queue manager, which can accept MQI calls directly from application programs that are running on the server processor; in addition, it can accept MQI requests from MQSeries clients.

This allows you to have an application that uses the MQI running on one machine, the client machine, and the queue manager itself running on a different machine.

Clients and servers can be useful in a number of situations:

- Where there is no full MQSeries implementation for the machine (for example, because it is a DOS platform)
- Where the client machine is too small, or of insufficient processing power, to run a full queue manager with good performance
- Where you want to allow the application program on the client processor to connect to multiple queue managers on different server processors
- Where you might want to reduce systems administration effort

You can run an MQSeries application in both a full MQSeries environment and in a MQSeries client environment without changing your code. However, the libraries you use at link-edit time determine the environment your application must run in.

When an application program in the client issues an MQI call, the client formats the parameter values of the call into an MQI request, and sends the request to the server. The server receives the request, performs the action specified in the request, and sends a response back to the client. The response is used by the client to generate information that is returned to the application program using the normal MQI return mechanism.

An additional function supported by MQSeries clients is the ability of an application program to be connected to more than one queue manager at a time, with the queue managers being on different processors or on the same processor.

Communication between clients and servers

An MQSeries client communicates with an MQSeries server, using an *MQI channel*, which is used to transfer MQI call requests from the client to the server, and responses from the server back to the client.

MQI channels differ from *message channels* (that are used to connect queue managers) in two ways:

- **An MQI channel is bidirectional.** One MQI channel can be used to send requests in one direction, and responses in the opposite direction.

  With message channels, data can be passed in one direction only. If two-way communication is required between two queue managers (for example, in the situation where reply messages are to be sent to the same queue manager that handled an initial request message), then two message channels are required, one to handle messages travelling in one direction, and another for messages travelling in the other direction.
- **Communication on an MQI channel is synchronous.** When an MQI request is transmitted from a client to a server, the MQSeries client product must wait for a response from the server before it can send the next MQI request.

  With message channels, the message traffic on the channel is time-independent. Multiple messages can be sent from one queue manager to the other, without the sending queue manager having to wait for any replies from the receiving queue manager.

The transmission protocol that is to be used on an MQI channel is specified as part of the channel definition. The MQSeries application program is unaware of the particular protocol that is being used on the channel. Furthermore, in the situation where an application program is connected to more than one MQSeries server, the MQI channels that are used for these connections could use different protocols. For example, an application program could connect to one queue manager using TCP on one channel, and to another queue manager using NetBIOS on a different channel.

With both MQI and message channels, a channel definition is required at each end of the channel, and each of these definitions must include a *channel type* and a *channel name*. You can choose to use different channel types according to the application you are designing, but the same channel name must be used at both ends of the channel.

Installing clients and servers

For those MQSeries platforms on which both servers and clients are supported, the MQSeries product includes the files for the MQSeries server and the MQSeries client for the server platform. In most cases, client files for a variety of other platforms are also provided.

For example, if you order MQSeries for AIX, in addition to the server product you receive files for the AIX client, plus the client code for DOS, HP-UX, OS/2 Warp, Sun Solaris, and various Windows platforms.

The client software can be loaded to memory in the client processor either from the disk on the client or server machine, or from a disk on a LAN file server.

You can install the client and the server either by installing from the media on which the products are supplied, to the client or server disk, or by installing to a LAN file server, and loading the client or server from there.

MQSeries for AS/400, MQSeries for OS/390, MQSeries for Tandem NSK, and MQSeries for VSE/ESA support client attach, but there is no client code for these platforms. Client code for other platforms can be obtained from the MQSeries Web site.

MQSeries clients from IBM Transaction Processing SupportPacs

You can install the MQSeries client files from an IBM Transaction Processing SupportPac™.

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.ibm.com/software/ts/mqseries/txppacs/txpsumm.html
```

MQSeries client software is available at no charge but 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.

The VM/ESA client is shipped with the VM/ESA product; it is not available as a SupportPac.

National language considerations for clients

The client part of an MQSeries product includes a file that contains all the program and operator messages that are used by the product. This file has been translated to the national languages of the server product, so that no further translation of these messages is required.

Data conversion considerations for clients

The data conversion facilities provided for MQSeries application programs are the same as those that are available for application programs that are executing with MQSeries servers:

- For certain built-in formats, conversion can be performed during the processing of an **MQGET** call, if the data conversion option is included in the call.

- For application defined formats, the conversion can be performed by a user-exit program, called during the processing of an **MQGET** call.

For further information about data conversion with MQSeries clients and servers, refer to the *MQSeries Application Programming Guide*.

Product support for MQSeries clients

The platform support for MQSeries clients and servers is as follows. Any of the MQSeries products listed is installed as a *Base product and Server* (*Base product and Client Attachment feature* on MQSeries for OS/390). These MQSeries products can accept connections from the MQSeries clients on the platforms listed, subject to differences in coded character set identifier (CCSID) and communications protocol.

Note: If you are using previous versions of MQSeries products, make sure that code conversion from the CCSID of your client is supported by the server. See the language support tables in the *MQSeries Application Programming Reference* manual.

The following MQSeries products:
- MQSeries for AIX, Version 5.1
- MQSeries for AS/400, Version 5.1
- MQSeries for AT&T GIS UNIX, Version 2.2
- MQSeries for Compaq (DIGITAL) OpenVMS, Version 2.2.1.1
- MQSeries for HP-UX, Version 5.1
- MQSeries for OS/2 Warp, Version 5.1
- MQSeries for OS/390, Version 2 Release 1
- MQSeries for SINIX and DC/OSx, Version 2.2
- MQSeries for Sun Solaris, Version 5.1
- MQSeries for Tandem NSK, Version 2.2.0.1
- MQSeries for VSE/ESA, Version 2.1
- MQSeries for Windows NT, Version 5.1
- MQSeries PRPQ (limited availability) products:
  - MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX), Version 2.2.1
  - MQSeries for Solaris on Intel, Version 5.0
  - MQSeries for Windows NT on DIGITAL Alpha (AXP), Version 2.0.1

can accept connections from MQSeries clients on:
- AIX
- AT&T GIS UNIX (this platform has become NCR UNIX)
- Compaq (DIGITAL) OpenVMS
- DIGITAL UNIX (Compaq Tru64 UNIX)
- DOS
- HP-UX
- OS/2 Warp
- SINIX and DC/OSx
- Sun Solaris
- VM/ESA
- Windows 3.1
- Windows 95
- Windows 98
- Windows NT

MQSeries clients on other platforms

Further MQSeries clients that can connect to the MQSeries products listed above are available through the Internet. For details of the platform and support status, see:

http://www.ibm.com/software/ts/mqseries/txppacs/txpsumm.html

Migrating from MQSeries Version 2 products to Version 5

The internal format of the channel definition table has been changed in the MQSeries Version 5 products for AIX, AS/400, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT.

A Version 5 client will continue to work with a Version 2 client channel definition table (by default, the client channel definition table is called AMQCLCHL.TAB). However, a Version 2 client cannot read a Version 5 client channel definition table. If your client channel definition table is produced by a Version 5 server, you must use Version 5 clients. In this case, reinstall the MQSeries client code from an MQSeries Version 5 server.

Applications on Version 5 clients

A Version 5 client can connect to all queue managers, non Version 5 as well as Version 5. If you are connecting to a non Version 5 queue manager you cannot use the new Version 5 features and structures in your MQSeries application on the client.

Where to find more information

More information about MQSeries clients can be found in the *MQSeries Clients* manual.

Chapter 8. Introduction to the MQSeries Framework

This chapter describes the MQSeries Framework. It contains the following sections:
- "Why the MQSeries Framework?"
- "Services and components provided" on page 56

Why the MQSeries Framework?

The MQSeries Framework offers users and independent software vendors the opportunity to extend or replace queue manager functionality, using defined interfaces.

These interfaces are provided in various forms. Some, for example the MQSeries name service interface, are provided by user-supplied modules that interface to the queue manager through an application programming interface. The trigger monitor interface, on the other hand, is provided by means of trigger messages that are written to a special queue.

In some cases components are shipped ready for you to use. You can choose whether to make use of them, and you can also decide to use your own versions instead of, or as well as, the supplied versions.

Not all MQSeries products provide all of the interfaces defined by the MQSeries framework. When an interface has been provided by an MQSeries product on a particular platform, it will be retained for future releases on that platform.

The major reasons for allowing modifications of the functions are:
- To provide the flexibility of choosing whether to use components provided by MQSeries products, or to replace or augment them with others
- To allow independent software vendors to participate, by providing components that might be using new technologies, without requiring internal changes to MQSeries products
- To allow MQSeries to exploit new technologies faster, and so provide products sooner

The components of the MQSeries Framework are:
- Trigger monitor interface (TMI)
- Message channel interface (MCI)
- Name service interface (NSI)
- Security enabling interface (SEI)
- Data conversion interface (DCI)

These are shown in Figure 9 on page 56, and are outlined in the following text. A detailed definition of each interface can be found in the book or books indicated.

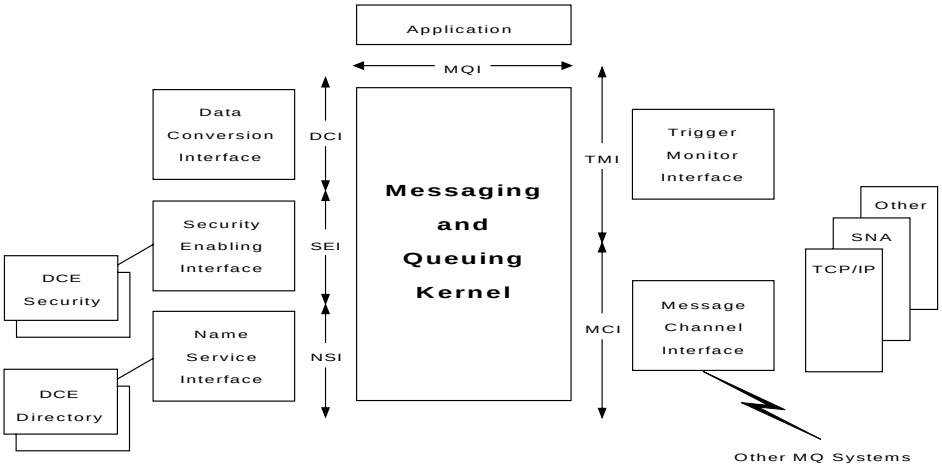

Figure 9. The MQSeries Framework

Invoking MQSeries Framework components

Where components are supported, they can be installed as necessary to provide some, or all, of the available function.

The queue manager refers to an initialization file to determine the particular components installed, and to get the information that it requires to invoke these components.

Services and components provided

This section gives information on the MQSeries Framework components and on the particular components that are provided by the MQSeries products.

Trigger monitor interface (TMI)

When messages arrive on a queue, the queue manager can generate trigger messages on a special kind of queue called an initiation queue. The generation of these messages is controlled by setting attributes of the queue.

The structure of the data in a trigger message is defined by the TMI. Trigger messages can be read by a long-running transaction called a trigger monitor, which starts applications to process the messages that have arrived on the queue.

MQSeries provides trigger monitors for use in various environments. When they start applications, they pass a defined structure as a parameter to the application. You can also provide your own trigger monitors, making use of the defined format of the trigger message. Your trigger monitors can pass the same structure as the supplied ones, or you can choose to do something different.

See the information on the trigger monitor interface (TMI) in the *MQSeries Application Programming Guide* for more information.

Message channel interface (MCI)

If a message is destined to go to a remote queue manager, the queue manager places it on a special kind of queue called a transmission queue. The message data has a header that includes the names of the destination queue manager and queue.

Triggering (see “Trigger monitor interface (TMI)” on page 56) can be used to start a process that reads the messages from a transmission queue, and sends them to their destination. Alternatively, this process can be started by an administrator.

MQSeries provides message channel agent (MCA) programs that transmit messages to MCAs on other queue managers. Commands are also provided to carry out channel administration.

You can provide alternative processes to take messages from transmission queues, and:

- Transmit the messages to similar processes you have provided on other queue managers, or
- Insert the messages into some other messaging system.

For details of how to implement the MCI, refer to the information on the transmission queue header structure (MQXQH) in the *MQSeries Application Programming Reference* manual to see how to write a message channel agent (MCA) program.

You can take advantage of triggering to initiate your process. You might need to provide administrative functions, similar to those available for MQSeries channels, for defining and monitoring the activity of your processes.

Note that the MCAs you write cannot communicate with MQSeries-supplied MCAs.

Name service interface (NSI)

Queue managers provide functions that allow administrators to define and manipulate queue definitions. Each queue manager maintains a directory of the queues that have been defined to it.

When an application opens a queue, the local queue manager looks up the queue in its directory if the application does not provide the name of a remote queue manager to which the queue belongs. If the queue is not found in that directory, the open normally fails.

However, you can install a naming service, which will be invoked if the queue manager is unable to resolve a queue name. Your service is passed the name of the queue that cannot be resolved. If your service recognizes the name, it returns the name of the queue manager to which the queue belongs.

The service provides the following functions:

- Lookup queue - given a queue name, finds the directory entry that contains that name, and returns the queue manager name
- Insert queue - inserts a new entry into the directory
- Delete queue - deletes an entry from the directory

The interface to the naming service is a programming interface, called the NSI. It is described in the *MQSeries Programmable System Management* manual.

The NSI is available on MQSeries for OS/2 Warp, MQSeries for Windows NT, and MQSeries on UNIX systems.

DCE naming component

The *DCE naming* installable component provides naming services for queues within one DCE cell.

Details of this component can be found in the *MQSeries Programmable System Management* manual.

The component is provided with MQSeries for AIX, Compaq (DIGITAL) OpenVMS, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT, but is not invoked unless it is included in the queue manager initialization file. The component requires that DCE support be available.

Security enabling interface (SEI)

There are three parts to the SEI:
- Authorization service
- User identifier service
- Message channel agent exits

See also "Chapter 3. Introduction to MQSeries security" on page 21 for further information.

Authorization service

This service provides access control facilities to the queue manager when an application issues an MQI call (for example, MQOPEN) or a command (for example, DELETE QLOCAL) that requires an authorization check to be carried out.

This enables the queue manager to check that users, or programs, have the appropriate authority for the actions they are trying to perform on queue manager objects.

The authorization service provides the following functions:
- Check object authority
- Set object authority
- Set initial authority
- Delete object authority
- Get object authority
- Copy all object authority

You can install your own version of this service, which can either be self-contained, or might in turn interface to some other authorization service that is available on the platform. Your service is invoked through a programming interface, which is part of the SEI. It maintains (either itself or by using another underlying service) lists of authorizations, and upon request returns information to the queue manager about whether a particular principal has authority to perform a certain action on a specified object.

This service is provided on MQSeries for AS/400, MQSeries for OS/2 Warp, MQSeries for Tandem NonStop Kernel, MQSeries for UNIX systems, and MQSeries for Windows NT.

Object authority manager: The *object authority manager (OAM)* is an installable component that provides authorization services.

The component is provided as part of the MQSeries products on the following platforms:
- Compaq (DIGITAL) OpenVMS

- OS/400
- Tandem NSK
- UNIX systems
- Windows NT

MQSeries for OS/2 Warp does not provide an authorization component, but is designed to accept any components (that provide authorization service functions) you might want to supply or obtain from independent software vendors.

User identifier service

On OS/2 Warp, it is not necessary for a user to log into the system, and therefore the queue manager cannot normally find a specific user identifier to associate with an application running on OS/2 Warp.

The user identifier service defines a programming interface, part of the SEI, which allows you to install your own service to supply a user identifier to the queue manager.

This service is provided only on MQSeries for OS/2 Warp.

This service is used by the queue manager to obtain a user ID. By default, the queue manager places the user ID in the message descriptor of messages when it puts them on queues, so that the application program that gets the messages can verify that they originated from authorized users or programs.

The queue manager can also make use of this user identifier for authorization checking.

The user identifier service provides just one function:

- Find user ID - obtain the predefined user ID

The user identifier service is not provided for client applications, for which another technique is available. See the *MQSeries Clients* manual.

Environment user ID: The *environment user ID* installable component provides the user identifier service, by obtaining a user ID value from an environment variable. The value of this variable is set when the component is installed.

Message channel agent exits

A mechanism for transmitting messages from one queue manager to another is referred to as a message channel. At each end of a channel a program called a message channel agent (MCA) controls the transmission of messages along that channel.

Exit points are defined at various points in the operation of an MCA. You can define functions that will be invoked at these points. Exit points are of particular relevance to security, and are therefore part of the SEI. They are:

- Security exits

  At each end, a security exit is invoked after communication has been established with the partner, but before message transfer starts. Each security exit has the opportunity to exchange security messages with the security exit at the partner, in order to satisfy itself of the partner's authenticity. If either is not satisfied, it can prevent any message transfer.
- Message exits

A message exit is invoked at the sending end just before a message is transferred, and at the receiving end just before it is stored. It has the opportunity to change any information in the message, including information in the message header such as the user identifier. This facility can be used to translate user identifiers on entering a new security domain or, for example, to set a blanket user identifier that has low authorization when receiving messages along a channel from an untrusted node. Message exits can also reject or reroute messages.

On AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT, you can call more than one message exit.

- Send and receive exits

  These are invoked just before any transmission is sent, and just after one is received. You can make use of this to provide encryption of sensitive data sent across open networks.

  On AIX, HP-UX, OS/2 Warp, OS/400, Sun Solaris, and Windows NT, you can call more than one send or receive exit.

You can define different exit modules for different channels.

See the *MQSeries Intercommunication* manual for further information about channel exits.

Data conversion interface (DCI)

Messages sent between platforms that normally use different encodings (for example, where integer fields are byte-swapped) and coded character set identifiers (CCSIDs) require conversion. An application getting a message from a queue can request that the queue manager converts the message data into the encoding and CCSID of its choice, which will normally be the standard ones in use on the platform on which it is running.

The message header contains information about the encoding and CCSID of the message data. It also contains a format name, which identifies the shape of the data. You can install exits, which have the same names as format names, to perform data conversions on the corresponding messages. These exits interface to the queue manager through a programming interface, called the DCI.

To help you provide these exits, MQSeries provides a utility that accepts a C language structure definition and generates C source code that can be built to provide an exit.

Built-in conversion is provided for MQSeries-defined standard formats.

The DCI is implemented on MQSeries for AS/400, MQSeries for Compaq (DIGITAL) OpenVMS, MQSeries for OS/2 Warp, MQSeries for OS/390, MQSeries for Tandem NonStop Kernel, MQSeries on UNIX systems, and MQSeries for Windows NT. When transmitting messages from a platform that does support it to one that does not, you can ask for this function to be carried out on all messages sent along a particular channel. This means that when the messages arrive at their destination, they are already in the standard encoding and CCSID for that platform.

See the *MQSeries Application Programming Guide* for information about data-conversion exit programs.

Part 2. Planning for MQSeries for AS/400

Chapter 9. Introduction to MQSeries for AS/400

MQSeries for AS/400 runs on an AS/400 machine capable of running OS/400 to support messaging and queuing applications.

This chapter includes basic information on:
- "Planning for MQSeries"
- "Support for R/3 with MQSeries" on page 65
- "Capacity-unit pricing" on page 65
- "Installing and setting up MQSeries" on page 65

For more information on the hardware and software environment needed by the MQSeries for AS/400 product, see "MQSeries for AS/400" on page 233.

Planning for MQSeries

This chapter helps you to plan for the introduction of MQSeries for AS/400 into your enterprise and introduces the items you must consider when doing this planning.

This chapter does not provide detailed information to enable you to perform the installation of MQSeries for AS/400. That information is provided in the *MQSeries for AS/400 V5.1 Quick Beginnings* book.

There are several stages in planning for the use of MQSeries for AS/400 that you must go through. They are:
1. Preparing your applications for the use of MQSeries for AS/400
2. Planning to include MQSeries for AS/400 in a network
3. Preparing to install MQSeries for AS/400
4. Planning to set up MQSeries for AS/400

A prime requirement for a message delivery system is that it must be reliable. Many functions are built into MQSeries for AS/400 to ensure that:
- Messages are not lost despite system failures.
- Messages are not delivered more than once.
- Messages are not accessed or delivered to unauthorized persons or applications.

MQSeries for AS/400 uses OS/400 journaling and other facilities to support these functions.

You must also plan for the operation and administration of MQSeries for AS/400 in your enterprise and consider the implementation of the MQSeries for AS/400 security facilities in addition to those of OS/400. Brief outlines of these planning operations are included in this chapter.

Preparing your applications for MQSeries

MQSeries for AS/400 brings the Message Queue Interface (MQI) to your applications. This interface allows you to modify existing applications and to write new applications. The MQI removes much of the need to understand the network and communication systems that you use. Thus you can expect to generate

applications more speedily than before. However, you must plan to take advantage of the MQI by planning its use in your applications and by understanding the ways in which it assists you.

You can find more information on how to use the MQI in your applications by referring to the *MQSeries Application Programming Guide*.

Interfacing with CICS

With the MQSeries for AS/400 products you can create application programs for the CICS for AS/400 transaction environment. These applications can use the MQI to communicate with CICS or non-CICS programs in any of the environments supported by the MQSeries products.

Using C++

IBM ILE C++ for AS/400 (program 5799-GDW) is a native compiler for C++ programs. In addition, IBM VisualAge® for C++ for AS/400 provides cross-compilers with clients running on OS/2 Warp, Windows 95, Windows 98, and Windows NT and generating object modules that can be bound into AS/400 programs.

Using Java™

A Java Virtual Machine is included in OS/400 V4R2, and later releases. For application development, the AS/400 Developer Kit for Java, 5769-JV1, and the OS/400 Qshell Interpreter, 5769-SS1 Option 30, are required.

Planning to use MQSeries in a network

MQSeries for AS/400 uses the distributed queuing facility to exchange messages between MQSeries platforms using the SNA LU 6.2 and TCP transmission protocols.

You must consider how you will attach MQSeries for AS/400 to a network, and how you will define the channels that are used to exchange messages.

According to the way that you have set your systems up, security checks can be performed at various times. MQSeries for AS/400 does not provide communications link authorization or data encryption on these links. Instead, various exits are provided that can be used by your applications to provide these facilities.

"MQSeries interoperability summary" on page 223 shows some of the links that are possible to other MQSeries products, and the transmission protocols that are used on these links. For further information about distributed queuing, refer to the *MQSeries Intercommunication* manual.

Planning recovery services with MQSeries

MQSeries for AS/400 makes use of the OS/400 journaling service to allow backup and recovery of the messaging system. "Chapter 10. Backup and recovery planning for MQSeries for AS/400" on page 67 introduces you to the recovery facilities and to the items you must consider in order to include MQSeries for AS/400 in your backup and recovery plans.

Planning data security with MQSeries

MQSeries for AS/400 uses the MQSeries Object Authority Manager (OAM) to manage access to MQSeries resources. For more information, see "Chapter 11. Security planning for MQSeries for AS/400" on page 69.

Administration of MQSeries

A summary of the administration facilities provided by the MQSeries for AS/400 products is given in "Chapter 12. Administration of MQSeries for AS/400" on page 73. Full details of these facilities can be found in the *MQSeries for AS/400 V5.1 System Administration* book.

Support for R/3 with MQSeries

The MQSeries link for R/3 for AS/400 product provides an interface that enables you to integrate your R/3 application with applications running in other environments (including those on R/3 and R/2 environments).

The R/3 link works with the Application Link Enabling (ALE) layer of the R/3 system to transmit Intermediate Documents (IDocs) into and out of your R/3 system, using MQSeries messages and queues to carry the information. It extends the scope of your business by allowing you to link your R/3 applications to any other application that you can access through MQSeries, even when those applications require different data formats.

For more information, see the *MQSeries link for R/3 Version 1.2 User's Guide*.

Capacity-unit pricing

The various processors on which MQSeries for AS/400 V5R1 can be installed are arranged in *processor group tiers*; the least powerful processors are in the lowest tier, the most powerful are in the highest. Associated with each tier is a number of *Capacity-Unit Use Authorizations* that must be purchased before MQSeries for AS/400 V5R1 can be used on any processor in that tier.

The minimum number of capacity units, which allows use of MQSeries for AS/400 V5R1 on an AS/400 P05 processor, is included in the licensed program packaging. To use MQSeries for AS/400 V5R1 on more powerful systems, or on additional systems, you must purchase additional capacity units.

The charge for MQSeries client access, where applicable, is included in the capacity unit charge.

Installing and setting up MQSeries

MQSeries for AS/400 is installed using the OS/400 RSTLICPGM command.

To prepare for the installation, you need to plan how much DASD you require in your OS/400 system to accommodate MQSeries for AS/400.

Assistance is given in "Chapter 13. Storage planning for MQSeries for AS/400" on page 77 to help you plan the amount of DASD required by MQSeries for AS/400.

Installing MQSeries

For MQSeries for AS/400, you must follow the instructions in the *MQSeries for AS/400 V5.1 Quick Beginnings* book.

The installation verification procedure is performed by creating and starting a queue manager, and creating the undelivered-message (dead-letter) queue.

Setting up MQSeries

MQSeries for AS/400 requires some setting up 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. Below are the items you must consider:

- Define queues.
  - Consider your naming conventions for queues.
- Define trigger processes.
- Define remote links.
  - Define associated transmission queues.
  - Consider your naming conventions for remote queues.
  - Consider your naming conventions for channels.

The setup procedures are described in the *MQSeries for AS/400 V5.1 System Administration* book.

Chapter 10. Backup and recovery planning for MQSeries for AS/400

MQSeries for AS/400 uses the journaling and recovery facilities provided by OS/400. You should be familiar with these facilities and refer to the *AS/400 Backup and Recovery* manual.

Each MQSeries for AS/400 queue manager has two unique journals that are created when the queue manager is created. These journals and the associated journal receivers are in the appropriate queue manager library.

MQSeries data events are recorded in these journals under the normal AS/400 controls. The events that are recorded are those that are required to recover MQSeries in case of failure. For message queues, all persistent messages are recorded in the journals; nonpersistent messages are not. This means that, in the event of a problem causing loss of messages that exist on a queue, the persistent contents can be recovered by use of the journals. The methods for recovery of such data or messages are discussed in the *MQSeries for AS/400 V5.1 System Administration* book.

You need to consider how the message data routed through MQSeries is to be journaled, and how you will extend your existing recovery plans to include MQSeries-related data.

Journal control with MQSeries

When data in a journal receiver reaches a threshold value, a new one is automatically created, attached, and brought into use by MQSeries. The old journal receiver is not deleted by MQSeries; this is a task that must be completed by the system administrator. MQSeries issues a message to the administrator containing some key dates that are associated with the journal receiver that has been replaced by the new one. It is up to your system administrator to decide what action must be taken to dispose of the old journal receiver using the dates supplied in the message. The action might be to move the old journal receiver to long term storage, or simply to delete it. However, deletion of journal receivers must be carefully considered as the records might still be needed for restart. The receivers might contain information regarding long-lived persistent messages. The choice is dependent on your strategy for data security and recovery.

Figure 10 on page 68 illustrates the concepts of journaling with MQSeries for AS/400 and shows how the message sent to the administrator contains date information pointers to indicate the journal receivers that should be saved to off-line media, those saved journal receivers that can be deleted, and those online journal receivers that can be freed. System backups have not been considered in the above description of journaling for MQSeries. You should continue to take system backups as required by your standard system operating procedures.

For more information on creating a system backup plan, refer to the *AS/400 Backup and Recovery* manual.

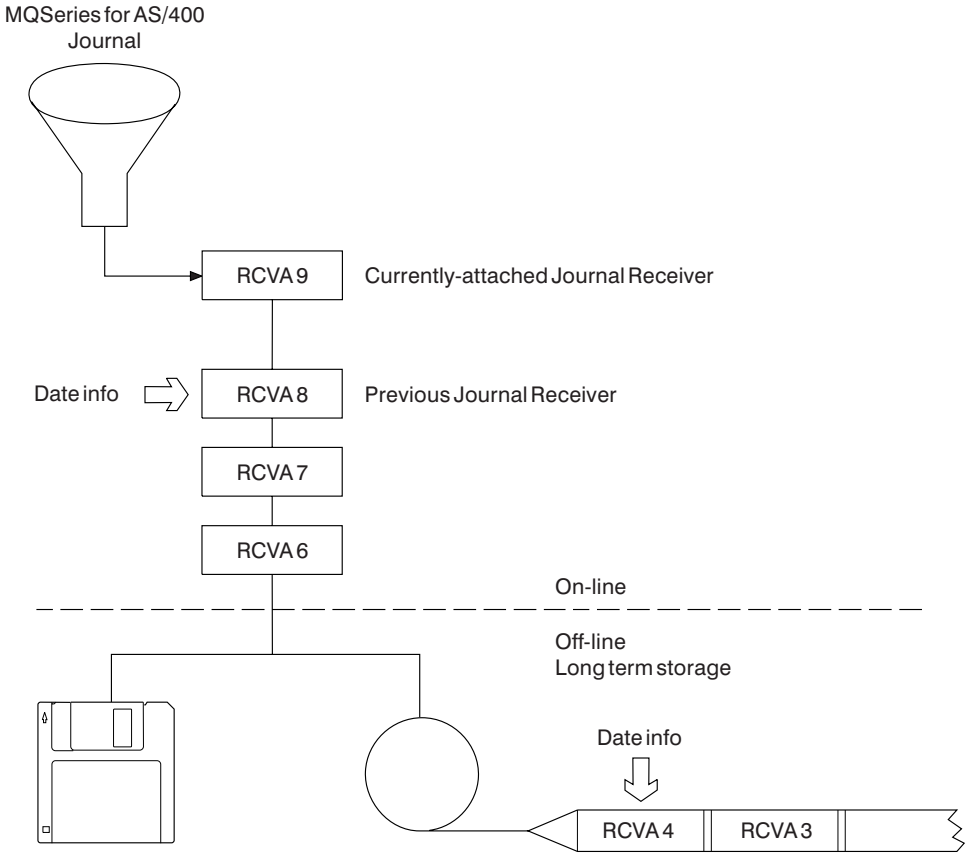

Figure 10. Journaling and date messages

Migration considerations

In MQSeries for AS/400 V5R1, there are journals and journal receivers for each queue manager in the queue manager library. In earlier versions of MQSeries for AS/400, the journals are in QUSRSYS and the receivers are in QMQMDATA. You must alter your backup and recovery strategy to reflect this difference if you are migrating to MQSeries for AS/400 V5R1 from an earlier version of the product.

Chapter 11. Security planning for MQSeries for AS/400

This chapter describes the access control security features in MQSeries for AS/400. It contains these sections:

- "Controlling access to resources with MQSeries"
- "Security exits with MQSeries" on page 71

Detailed information about MQSeries for AS/400 security handling is given in the *MQSeries for AS/400 V5.1 System Administration* book.

Controlling access to resources with MQSeries

With MQSeries for AS/400, access to queue manager resources is controlled through the *object authority manager (OAM)*, which is the default authorization service installable component. Because the OAM is an installable component, you can implement your own security controls in place of, or in addition to, those supplied by the OAM. For more information about installable services and installable components, see "Chapter 8. Introduction to the MQSeries Framework" on page 55.

Users can access queue manager objects (queues, process definitions, namelists, channels, and queue managers) only if they have the required authority. The OAM manages a user's authorization to manipulate MQSeries objects, and provides a command interface through which you can grant or revoke access authority to an object for a specific group of users.

Managing access through user groups with MQSeries

In discussing security in an MQSeries for AS/400 environment, we use the term *principal* rather than user ID. The reason for this is that authorities granted to a user ID can also be granted to other entities, for example, an application program that issues MQI calls, or an administration program that issues PCF commands. In these cases, the principal associated with a program is not necessarily the user ID that was used when the program was started.

Managing access permissions to MQSeries resources is based on *user groups*, that is, groups of principals. The OAM does not maintain authorizations at the level of individual principals. The mapping of principals to group names is carried out within the OAM, and operations are carried out at the group level.

The authorizations that a principal has are the union of the authorizations of all the groups of which it is a member, that is, its *group set*. Whenever a principal requests access to a resource, the OAM computes this union, and then checks the authorization against it.

Resources you can protect with MQSeries

Through MQSeries for AS/400 you can control:

- **Access to queue manager objects through the MQI**

  When an application program attempts to access an object, the OAM checks that the principal making the request has the authorization (through its user group) for the requested operation. In this way, the queues, and the messages on queues, can be protected from unauthorized access.

- **Permission to use MQSeries commands**

  Only members of authorized user groups can execute queue manager administration commands.

  When a member of a user group attempts to execute a command, the OAM checks that the principal making the request has the authorization (through its user group) to use the command. These access checks are performed irrespective of the interface used (CL, MQSC, PCF, or MQSeries control commands).

Different groups of users can be granted different kinds of access authority to the same object. For example, one group might be allowed to read (**MQGET**) messages from a queue, but not to write (**MQPUT**) messages to that queue. Other groups might be given authority to put messages to and get messages from the same queue. Similarly, some groups might have get and put authority but not be allowed to create or delete queues.

Using groups for authorizations with MQSeries

Using groups for authorization, rather than individual principals, reduces the amount of administration required. Normally, a particular kind of access is required by more than one principal. For example, you might define a group consisting of end users who want to run a particular application. New users can be given access simply by adding their user ID to the appropriate group.

Try to keep the number of groups as small as possible. Dividing principals into one group for application users, and one for administrators, is a good place to start.

MQSeries for AS/400 user profiles

MQSeries for AS/400 supplies two user profiles:

QMQM
: User profile QMQM owns all MQSeries objects, including files in the Integrated File System (IFS). All MQSeries jobs run under user QMQM.

QMQMADM
: QMQMADM is the administration profile. QMQMADM is the primary group for MQSeries for AS/400, and is the group profile for QMQM. Any user who requires administrative access to MQSeries for AS/400 must belong to group QMQMADM.

Using the security commands with MQSeries

The OAM provides three CL commands that you can use to manage the authorizations of users. These are:

GRTMQMAUT
Grant authority.

RVKMQMAUT
Revoke authority.

DSPMQMAUT
View authority.

When you start a queue manager, all associated jobs run under user ID QMQM. After this, any principal belonging to the group QMQMADM can issue GRTMQMAUT and RVKMQMAUT commands to change authorizations to resources.

For more information, see the *MQSeries for AS/400 V5.1 System Administration* book.

Security exits with MQSeries

The message channels that are used for distributed queuing, and the MQI channels that are used between clients and servers, both have security exit facilities that can invoke programs that you have supplied.

For more information on these security exit programs, see the *MQSeries Intercommunication* manual.

Chapter 12. Administration of MQSeries for AS/400

This chapter is a summary of the administration facilities provided by the MQSeries for AS/400 products. The chapter has the following sections:

- "Managing objects with MQSeries"
- "Remote administration with MQSeries" on page 74

Details of the commands, command interfaces, and utilities that are provided by MQSeries for AS/400 are given in the *MQSeries for AS/400 V5.1 System Administration* book.

Managing objects with MQSeries

It is the administrator's job to monitor MQSeries for AS/400 and make any changes that might be necessary. To do this, the administrator needs to know where each MQSeries object resides, what its characteristics are, and who has access to it.

The administrator can manage and monitor MQSeries resources interactively using MQSC or OS/400 CL commands. Alternatively, if you have sets of commands that you issue regularly, you can write a CL program or MQSC script. Information about writing administration programs is in the *MQSeries for AS/400 V5.1 System Administration* book. You can also use the MQSeries Explorer application, which is part of MQSeries for Windows NT V5.1, to administer MQSeries for AS/400 remotely.

MQSeries for AS/400 can ensure that the user is authorized to issue particular commands for particular resources.

Commands on MQSeries

MQSeries for AS/400 supports the following administration facilities:

- **CL Commands**

  MQSeries for AS/400 provides Control Language commands (CL commands). These can be issued either at the command line, or by writing a CL program.

  Examples of CL tasks include:
  - Authorize system administrators.
  - Start and stop a queue manager.
  - Define, change, and display a queue.
  - Define, change, and display a process.
  - Define, test, and delete a channel.

  CL commands are designed exclusively for OS/400, and CL responses are designed to be human-readable. The commands perform similar functions to PCF commands, but the format is designed to match the standard OS/400 format, whereas PCF commands are platform independent, with both the command and response formats being intended for program use.

- **MQSeries commands (MQSC)**

  With MQSeries for AS/400, you can build commands by entering the MQSC commands into a member of a specified library and running the STRMQMMQSC command. For information about how to do this, see the *MQSeries Command Reference* manual.

- **PCF commands**

  Any local or remote application program can generate PCF commands in messages and put them to the command queue, SYSTEM.ADMIN.COMMAND.QUEUE, to be processed by the MQSeries for AS/400 command server.

More information about using these facilities is given in the *MQSeries for AS/400 V5.1 System Administration* book.

MQSeries for AS/400 can verify that the user is authorized to issue particular commands for particular resources.

Changes made to the resource definitions of a queue manager using the commands (directly or indirectly) are preserved across restarts of the MQSeries for AS/400 system.

Managing communications with MQSeries

Part of the administrator's role is to ensure that the required communications links are activated, and to monitor the status of these links as required by your enterprise. You can find information describing these tasks in the *MQSeries Intercommunication* manual.

Remote administration with MQSeries

There are two aspects to the MQSeries remote administration facilities:

- Using MQSeries for AS/400 to manage remote MQSeries systems
- Using other MQSeries systems to manage MQSeries for AS/400

Managing remote systems from MQSeries

To manage remote MQSeries systems from MQSeries for AS/400, you can:

- Write an application program to generate PCF commands and put them to the command queue at the remote queue manager. (Note that some products do not have a command queue, so cannot be administered from a local or remote application program.)
- Issue MQSC commands in indirect mode.

Because of the differences in the MQSeries products, it is not always possible to manage remotely the same set of MQSeries objects or attributes that you can manage locally.

Managing MQSeries from remote systems

MQSeries for AS/400 can be managed from a remote MQSeries system, by an administrator using the facilities provided by the following products:

- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/2 Warp
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Notes:

1. You can manage MQSeries for AS/400 from MQSeries for OS/390 by writing an application program to send the appropriate PCF commands to the command queue.
2. The MQSeries for Windows NT V5.1 MQSeries Explorer application is an interactive, graphical tool that can be used for administration of MQSeries for AS/400 systems.

Chapter 13. Storage planning for MQSeries for AS/400

This chapter tells you how to plan the type and amount of storage you require when you include MQSeries for AS/400 in your system. It contains information about:

Product storage for MQSeries

MQSeries for AS/400 requires approximately 100 MB of disk storage in the QMQM library.

This size includes the product programs, message files, panels and prompts, commands, and ancillaries. The size varies slightly according to the options you install.

See the *MQSeries for AS/400 V5.1 System Administration* book for details of the available options.

Journal storage for MQSeries

Significant events and data changes are journaled by OS/400. Such data is sent by MQSeries for AS/400 to unique journals. These journals direct the data to a series of predefined journal receivers. The threshold values of these receivers are set up by MQSeries for AS/400 to a default size of 16 MB.

When data in a journal receiver reaches its threshold value, a new one is automatically created, attached, and brought into use by MQSeries for AS/400. A message is sent to the system administrator containing key dates associated with the journal receivers. Action must be taken by the administrator to dispose of the old journal receiver according to the dates supplied in the message. See "Chapter 10. Backup and recovery planning for MQSeries for AS/400" on page 67 for more information about journaling.

You need to plan for some permanent storage to which journal receivers can be offloaded when they are no longer required to be maintained in main storage. The journals can be offloaded to diskettes, DASD, tape, or other media supported in your enterprise.

Storage for other data sets used by MQSeries

The total amount of storage you should reserve for MQSeries for AS/400 depends mainly on the amount needed to store messages; in the majority of cases, the amount needed to store the other MQSeries objects that are required, such as process objects, and the queue manager object, is small compared to that required for messages.

Therefore, you can get an approximate figure for the amount of storage required in the data sets, by calculating the amount required for both persistent and nonpersistent messages, using the algorithms described in the next section, and adding the two amounts together.

Message queues for MQSeries

Messages are stored in the queue manager's directory in the Integrated File System (IFS). To estimate the total amount of storage you will need for queues, you need to know:

- How many queues you have.
- The maximum number of messages there will be on each queue at any one time.
- The average message size for each queue. The amount of storage required for one message is the size of the message data plus the size of the message header (456 bytes), rounded up to the nearest 512-byte block. If you are using distribution lists, or grouped or segmented messages, the size of the header will increase for the transmission queue.

Given these values, you can calculate the total amount of space required for queues. However, the value you calculate is likely to be approximate only. You are recommended to add a contingency amount to avoid shortage of queue space when an application is running.

The maximum size of a single queue on MQSeries for AS/400 is 2 GB.

Performance information for MQSeries

Information about MQSeries performance is available on the Internet at:

`http://www.ibm.com/software/ts/mqseries/txppacs/txpm1.html`

Part 3. Planning for MQSeries for Compaq (DIGITAL) OpenVMS

Chapter 14. Introduction to MQSeries for Compaq (DIGITAL) OpenVMS

There are two MQSeries products for Compaq (DIGITAL) OpenVMS; one for AXP and one for VAX. The products comprise two parts, the *server* and the *clients*. The server runs on a machine that is capable of running an MQSeries queue manager; the clients provided with the product are for Compaq (DIGITAL) OpenVMS, DOS, OS/2 Warp, and Windows 3.1. (The Windows 3.1 client can operate under Windows 3.1, Windows 95 or within the WIN-OS/2® environment under OS/2 Warp.)

MQSeries provides the MQI programming interface for use by application programs that are running on the server or the client processor. More detail on MQSeries clients and servers is given in "Chapter 7. Introduction to MQSeries clients and servers" on page 49.

For information on the hardware and software environments, see "MQSeries for Compaq (DIGITAL) OpenVMS" on page 235.

Planning for MQSeries

This chapter helps you to plan for the introduction of MQSeries for Compaq (DIGITAL) OpenVMS into your enterprise, and introduces the items that you need to consider when doing this planning.

There are several stages in planning for the use of MQSeries for Compaq (DIGITAL) OpenVMS that you must go through. They are:

1. Preparing your applications for the use of MQSeries for Compaq (DIGITAL) OpenVMS
2. Planning to include MQSeries for Compaq (DIGITAL) OpenVMS in a network
3. Preparing to install MQSeries for Compaq (DIGITAL) OpenVMS
4. Planning to set up MQSeries for Compaq (DIGITAL) OpenVMS

A prime requirement for a message delivery system is that it must be reliable. Many functions are built into MQSeries for Compaq (DIGITAL) OpenVMS to ensure that:

- Messages are not lost despite system failures.
- Messages are not delivered more than once.
- Messages are not accessed by, or delivered to, unauthorized persons or applications.

MQSeries for Compaq (DIGITAL) OpenVMS uses logging and other facilities to support these functions.

You must also plan for the operation and administration of MQSeries for Compaq (DIGITAL) OpenVMS in your enterprise, and consider the implementation of an appropriate set of security facilities. Brief outlines of these planning operations are included in this chapter.

Preparing your applications for use with MQSeries

MQSeries for Compaq (DIGITAL) OpenVMS brings the Message Queue Interface (MQI) to your applications. This interface allows you to modify existing

applications and to write new applications. The MQI removes much of the need to understand the network and communication systems that you use. Thus you can expect to generate applications more speedily than before. However, you must prepare to take advantage of the MQI by planning its use in your applications and by understanding the ways in which it assists you.

You can find more information on how to use the MQI in your applications by referring to the *MQSeries Application Programming Guide*.

Planning to use MQSeries in a network

MQSeries for Compaq (DIGITAL) OpenVMS uses the distributed queue management (DQM) facility to exchange messages between MQSeries platforms, using either the DECnet, SNA LU 6.2 or TCP transmission protocols.

You must consider how you will attach MQSeries for Compaq (DIGITAL) OpenVMS to a network, and how you will define the message channels that will be used to exchange messages.

According to the way that you have set your systems up, security checks can be performed at various times. Various exits are provided that can be used by your applications to provide these facilities.

"MQSeries interoperability summary" on page 223 shows the links that are possible to other MQSeries products, and the transmission protocols that can be used.

For further information about distributed queue management, refer to the *MQSeries Intercommunication* manual.

Installing MQSeries

MQSeries for Compaq (DIGITAL) OpenVMS is installed with the OpenVMS VMSINSTALL utility. For information about installing MQSeries, see the *MQSeries for Digital OpenVMS System Management Guide*.

To prepare for the actual installation, you need to plan how much disk space will be required in your Digital OpenVMS system to accommodate MQSeries. Assistance is given in "Chapter 18. Storage planning for MQSeries for Compaq (DIGITAL) OpenVMS" on page 95 to help you plan the amount of space required.

Setting up MQSeries

After installation, MQSeries for Compaq (DIGITAL) OpenVMS needs to be set up, and customized for your own use. This ensures that the appropriate Digital OpenVMS facilities are made available to MQSeries, and that your MQSeries system is correctly initialized and ready to work with your applications.

MQSeries for Compaq (DIGITAL) OpenVMS uses configuration files to hold the product configuration information used for logging, communications protocols and installable components. After installing the product, you can edit these files to tailor the operation of the product to meet the requirements of your installation.

In addition, you need to do the following:

- Define queues.
  - Consider your naming conventions for queues.
- Define trigger processes.
- Define remote links.

- Define associated transmission queues.
- Consider your naming conventions for remote queues.
- Consider your naming conventions for channels.

Note: The characters within the names given to all MQSeries objects are case sensitive. Therefore, be very careful when defining the names of objects, to select the appropriate uppercase or lowercase characters.

You can find more information on the setting up and customizing processes for MQSeries for Compaq (DIGITAL) OpenVMS in the *MQSeries for Digital OpenVMS System Management Guide*.

Planning recovery services with MQSeries

MQSeries for Compaq (DIGITAL) OpenVMS provides logging services to allow backup and recovery of the messaging system. “Chapter 15. Backup and recovery planning for MQSeries for Compaq (DIGITAL) OpenVMS” on page 85 introduces you to these facilities, and to the items that you need to consider in order to include MQSeries for Compaq (DIGITAL) OpenVMS in your backup and recovery plans.

You can find more information on the backup and recovery facilities provided by MQSeries for Compaq (DIGITAL) OpenVMS in the *MQSeries for Digital OpenVMS System Management Guide*.

Planning data security on MQSeries

MQSeries for Compaq (DIGITAL) OpenVMS uses the facilities of the MQSeries *object authority manager (OAM)* installable component to control access to the various different types of queue manager resource (queues, process definitions, channels, and queue managers).

You can find more general information on authorization installable components in “Chapter 8. Introduction to the MQSeries Framework” on page 55.

“Chapter 16. Security planning for MQSeries for Compaq (DIGITAL) OpenVMS” on page 89 introduces you to the security facilities provided by MQSeries for Compaq (DIGITAL) OpenVMS and to some of the items you need to consider when planning for security.

You can find more information on the security facilities provided by MQSeries for Compaq (DIGITAL) OpenVMS in the *MQSeries for Digital OpenVMS System Management Guide*.

Administration with MQSeries

A summary of the administration facilities provided is given in “Chapter 17. Administration of MQSeries for Compaq (DIGITAL) OpenVMS” on page 93. Full details of these facilities can be found in the *MQSeries for Digital OpenVMS System Management Guide*.

See also “MQSeries product administration facilities” on page 36.

Migration from MQSeries for Digital OpenVMS Version 1

To use MQSeries application programs that were written for MQSeries Version 1 with MQSeries for Compaq (DIGITAL) OpenVMS, you need to do the following:

1. Redefine all message queues.
2. Redefine all message channels.
3. Recompile the application programs, using the MQSeries Version 2 header files.

This might be a suitable time to consider whether you need to redesign any parts of your application, to take advantage of the additional function provided by MQSeries for Compaq (DIGITAL) OpenVMS.

One difference between MQSeries for Compaq (DIGITAL) OpenVMS Version 2 and the earlier products is that MQSeries for Compaq (DIGITAL) OpenVMS does its own queue storage management. It is not necessary to run a utility program to recover the space that was occupied by messages that have been removed from queues by MQGET calls; this is done automatically by MQSeries.

Chapter 15. Backup and recovery planning for MQSeries for Compaq (DIGITAL) OpenVMS

This chapter describes the background concepts of recovery and restart. It contains the following sections:
- "Logging with MQSeries"
- "Recovering from problems with MQSeries" on page 87

More details of the logging and recovery facilities of MQSeries for Compaq (DIGITAL) OpenVMS are given in the *MQSeries for Digital OpenVMS System Management Guide*.

Logging with MQSeries

The basic premise of a messaging system is that messages entered into the system are assured of delivery to the destination. One of the ways of ensuring that messages are not lost is to maintain a record of the activities of the queue manager that handles the receipt, transmission, and delivery of messages.

MQSeries for Compaq (DIGITAL) OpenVMS does this by recording all the significant changes to the data controlled or managed by the queue manager in a log. This process is called logging. The data changes that are logged include the puts and gets of persistent messages to and from queues, changes to queue attributes, and channel activity.

The purpose of logging is to create and maintain a log that:
- Keeps records of queue manager changes
- Keeps records of queue updates for use by the restart process
- Is a source for restoration of data should there be a hardware or software failure

Each MQSeries log consists of a log control file, together with one or more log files for the storage of data.

The log control file is, as its name implies, used to control and monitor the use of the log files. It contains information relating to the size, location, next available file, and other data related particularly to the log files themselves. All the log files within one log are the same size; there is a default value for this size, but you can override this value when you set up the log.

Types of logging for MQSeries

MQSeries for Compaq (DIGITAL) OpenVMS has two approaches to maintaining records of queue manager activities:
- Circular logging
- Linear logging

Each type of logging stores the recorded data in a set of files. The differences between the two types of logging are the contents, and the way that the files are linked together.

With circular logging, the set of log files are effectively linked together so as to form a ring. When data is collected, it is written sequentially into the files, in such a way as to reuse the log files in the ring. You can use circular logging for:

- Crash recovery - that is, after a system failure of some kind has stopped the queue manager unexpectedly
- Restart recovery - after a planned closedown of the system

With linear logging, the log is maintained as a continuous sequence of files. When data is collected, it is written sequentially into the log files; the space in the files is not reused, so that you can always retrieve any record from the time that the queue manager was created.

Because disk space is finite, you might have to plan for some form of archiving. Also, if you are handling a high volume of persistent messages, all your log files will eventually be filled. This will result in operator messages being written to an error log file; some action will need to be taken by the system administrator to make more log space available, or to reuse the existing space. You can use linear logging for:

- Crash recovery
- Restart recovery
- Media recovery - to recreate lost or damaged data after a media failure by replaying the contents of the log

Selecting a logging method with MQSeries

You must base your selection of log type on your requirements for recovery.

Both types of logging can cope with unexpected power outages in the absence of hardware failure. If you accept that only crash or restart recovery is required, circular logging might be adequate. If media recovery is important to you, select linear logging.

With each type of logging, you need to decide on the number of files to use in the log, and their size. The total amount of space needed depends on the amount of data to be recorded, which depends on various parameters, including:

- The size of messages
- The number of puts and gets from queues
- The number of messages being transmitted by the message channel agents

Recovering from problems with MQSeries

MQSeries can recover from communications failures and power loss incidents. In addition, it is sometimes possible to recover from other types of problem with the MQSeries data, such as inadvertent deletion of a file.

In the case of a communications failure, messages remain on the queues until they are removed by a receiving application. If the message is being transmitted, it remains on the transmission queue until it can be successfully transmitted. To recover from a communications failure, it is normally sufficient simply to restart the channels using the link that failed.

On a restart after your system has lost power, the queue manager restores all the persistent messages that were on the queues to the state that existed just before the power failure, so that no persistent messages are lost; nonpersistent messages are discarded.

There are ways in which an MQSeries object can become unusable, for example due to inadvertent damage. In such situations, you will have to take steps to recover either your complete system or some part of it. The action required depends on when the damage is detected, whether the log method selected supports media recovery, and which object or objects are damaged.

Chapter 16. Security planning for MQSeries for Compaq (DIGITAL) OpenVMS

This chapter describes the access control security features in MQSeries for Compaq (DIGITAL) OpenVMS. It contains these sections:
- "Controlling access to resources for MQSeries"
- "Security exits with MQSeries" on page 91

Full details of MQSeries for Compaq (DIGITAL) OpenVMS security handling are given in the *MQSeries for Digital OpenVMS System Management Guide*.

Controlling access to resources for MQSeries

With MQSeries for Compaq (DIGITAL) OpenVMS, access to queue manager resources is controlled through the *object authority manager (OAM)*, which is the default authorization installable component. Because the OAM is an installable component, you can implement your own security controls in place of, or in addition to, those supplied by the OAM. (For more information on installable services and installable components, see "Chapter 8. Introduction to the MQSeries Framework" on page 55.)

Users can access queue manager objects (queues, process definitions, channels, and queue managers) only if they have the required authority. The OAM manages a user's authorization to manipulate MQSeries objects, and provides a command interface through which you can grant or revoke access authority to an object for a specific group of users.

Managing access through user groups with MQSeries

In discussing security in a Compaq (DIGITAL) OpenVMS environment, we use the term *principal* rather than user ID. The reason for this is that authorities granted to a user ID can also be granted to other entities, for example, an application program that issues MQI calls, or an administration program that issues PCF commands. In these cases, the principal associated with a program is not necessarily the user ID that was used when the program was started.

Managing access permissions to MQSeries resources is based on Compaq (DIGITAL) OpenVMS *rights identifiers*, that is, identifiers held by principals. A principal can hold one or more OpenVMS rights identifiers. A group is defined as the set of all principals that have been granted a specific rights identifier.

The OAM does not maintain authorizations at the level of individual principals. The mapping of principals to identifier names is carried out within the OAM, and operations are carried out at the rights identifier level.

The authorizations that a principal has are the union of the authorizations of all the rights identifiers that it holds, that is, its process rights. Whenever a principal requests access to a resource, the OAM computes this union, and then checks the authorization against it.

Resources you can protect with MQSeries

Through MQSeries for Compaq (DIGITAL) OpenVMS you can control:

- Access to MQSeries objects through the MQI. When an application program attempts to access an object, the OAM checks if the user ID making the request has the authorization (through the identifier held) for the operation requested.

  In particular, this means that queues, and the messages on queues, can be protected from unauthorized access.
- Permission to use MQSC commands. Only principals that hold rights identifier **mqm** can execute queue manager administration commands, for example, to create a queue.
- Permission to use control commands. Only principals that hold rights identifier **mqm** can execute control commands, for example, creating a queue manager, starting a command server, or using runmqsc.
- Permission to use PCF commands.

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

Using rights identifiers for authorizations with MQSeries

Using identifiers for authorization, rather than individual principals, reduces the amount of administration required. Typically, a particular kind of access is required by more than one principal. For example, you might define a group consisting of end users who want to run a particular application. New users can be given access simply by granting the appropriate identifier to their OpenVMS user ID.

Try to keep the number of groups as small as possible. Dividing principals into one group for application users, and one for administrators, is a good place to start.

Using the security commands MQSeries

The OAM provides a command interface for granting and revoking authority. Before you can use these commands, you must be suitably authorized - your user ID must hold the OpenVMS rights identifier **mqm**. This identifier should have been set up when you installed the product.

If your user ID holds identifier **mqm**, you have a ‘super user’ authority to the queue manager. This means that you are authorized to issue any MQI request or command from your user ID.

The OAM provides two commands that you can invoke from your OpenVMS DCL to manage the authorizations of users. These are:

- **setmqaut** (set or reset authority)
- **dspmqaut** (display authority)

Details of these commands can be found in the *MQSeries for Digital OpenVMS System Management Guide*.

Security exits with MQSeries

The message channels that are used for distributed queuing, and the MQI channels that are used between clients and servers, both have security exit facilities that can invoke programs that you have supplied.

For more information on these security exit programs, see the *MQSeries Intercommunication* manual.

Chapter 17. Administration of MQSeries for Compaq (DIGITAL) OpenVMS

This chapter is a summary of the administration facilities provided by MQSeries for Compaq (DIGITAL) OpenVMS. It has the following sections:

- "Managing objects with MQSeries"
- "Remote administration with MQSeries" on page 94

Details of the commands, command interfaces, and utilities that are provided by MQSeries for Compaq (DIGITAL) OpenVMS are given in the *MQSeries for Digital OpenVMS System Management Guide*. You need to arrange for users who need to be able to use these administration facilities to have the necessary authorizations, using the procedures given in the *System Management Guide*.

Managing objects with MQSeries

It is the administrator's job to monitor MQSeries for Compaq (DIGITAL) OpenVMS and make any changes that might be necessary. To do this, the administrator needs to know where each MQSeries object resides, what its characteristics are, and who has access to it.

The administrator can manage and monitor the resources using MQSeries commands (MQSC), or, if there are sets of commands that are issued regularly, by writing an application program that places them on the command queue.

MQSeries can use the security features provided by the OAM, or by a security component that you have installed, to ensure that the user is authorized to issue particular commands for particular resources.

Commands on MQSeries

MQSeries for Compaq (DIGITAL) OpenVMS supports the following administration commands and facilities:

- You can enter control commands on the command line.
- You can use the **runmqsc** control command to cause MQSC commands from standard input to be executed.
- Any local or remote MQSeries application program can generate PCF commands in messages and put them to the command queue SYSTEM.ADMIN.COMMAND.QUEUE, to be processed by the MQSeries for Compaq (DIGITAL) OpenVMS command server.

More information on how to use all these facilities is given in the *MQSeries for Digital OpenVMS System Management Guide*.

Changes made to the resource definitions of a queue manager using the commands (directly or indirectly) are preserved across restarts of MQSeries for Compaq (DIGITAL) OpenVMS.

Managing communications with MQSeries

Part of the administrator's role is to ensure that the required communications links are activated, and to monitor the status of these links as required by your enterprise. You can find information describing these tasks in the *MQSeries Intercommunication* manual.

Remote administration with MQSeries

There are two aspects to the MQSeries remote administration facilities:

- MQSeries for Compaq (DIGITAL) OpenVMS can be used to manage remote systems.
- Other remote products can be used to manage MQSeries for Compaq (DIGITAL) OpenVMS.

Managing remote systems from MQSeries

Facilities are provided by MQSeries for Compaq (DIGITAL) OpenVMS to allow an administrator to manage the following remote systems:

- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/2 Warp
- MQSeries for OS/390
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Because of the differences in the MQSeries products, it is not always possible to manage remotely the same set of MQSeries objects or attributes that you can manage locally.

If you want to manage any other MQSeries product, you can write an application program to send the appropriate commands to the command queue for that product. However, some MQSeries products do not have a command queue, so they cannot accept commands from local or remote application programs.

Managing MQSeries from remote systems

MQSeries for Compaq (DIGITAL) OpenVMS can be managed from a remote MQSeries system, by an administrator using the facilities provided by the following products:

- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/390
- MQSeries for OS/2 Warp
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Note: You can manage MQSeries for Compaq (DIGITAL) OpenVMS from MQSeries for OS/390 by writing an application program to send the appropriate PCF commands to the command queue.

Chapter 18. Storage planning for MQSeries for Compaq (DIGITAL) OpenVMS

This chapter tells you how to plan the type and amount of storage you require when you include MQSeries for Compaq (DIGITAL) OpenVMS in your network. It has the following sections:

RAM considerations for MQSeries

The processor memory (RAM) is used by MQSeries for Compaq (DIGITAL) OpenVMS in the execution of the product modules, and as a paging area for the messages that are being processed. Parts of the paging area are written to, and read from, disk as necessary.

The minimum amount of RAM required to run the MQSeries for Compaq (DIGITAL) OpenVMS server is 16 MB on VAX and 32 MB on AXP, if more RAM is available, the performance of the message processing improves.

The amount of RAM required on each client system for an MQSeries client is small compared to that required for the operating system on each of the platforms.

Disk space considerations for MQSeries

Disk space is used by MQSeries for Compaq (DIGITAL) OpenVMS for the following:

- Product modules - client and server executable modules and the toolkit
- Paging space - server only
- Message queues - server only
- Logs - server only

Product modules for MQSeries

The disk space that you require for the product modules depends on the options that you decide to install: the options are described in the *MQSeries for Digital OpenVMS System Management Guide*.

Space might be required for client and server executable program modules and the toolkit. If all options are selected, 16 MB are required on VAX and 18 MB on AXP.

Paging space for MQSeries

The messages that the server is processing are stored in memory, and are paged to disk.

The algorithm used to calculate the size of the paging space required to support a particular application is complex, and is not given in this manual. Instead, you should refer to “Sample configuration for MQSeries” on page 97, which gives the paging space used on the typical minimum configuration.

Message queues for MQSeries

In order to estimate the total amount of storage that you will need for queues, you need to know:

- The number of queues that you have.
- The maximum number of messages there will be on each of the queues at any one time.
- The average size of message on each of the queues. The amount of storage required for one message varies. It is based on the size of the message data plus the size of the message header (456 bytes), rounded up the nearest 512-byte block. If you are using distribution lists, or grouped or segmented messages, the size of the header will increase for the transmission queue.

Given these values, you can calculate the total amount of space required for queues. However, this value is likely to be an approximate value only, and it is advisable to add a contingency value, to avoid the situation where there is no space left for messages on the queues when your application is running.

The maximum size of a single queue on MQSeries for Compaq (DIGITAL) OpenVMS is 320 MB.

Log files for MQSeries

Significant events and data changes can be logged in circular or sequential logs. In particular, the logs are used for recording persistent messages.

All the log files in a log are of the same size. By default, this size is 4 MB, but this value can be changed by the system administrator when the log is defined.

For a circular log, the system administrator needs to specify how many files should be included in the log. For a sequential log, the number of files will increase over time, until the system administrator archives some of the files, and disposes of them. You need to plan for the permanent storage (diskettes, tape, or other media supported in your enterprise) that is to be used for these archived files.

Sample configuration for MQSeries

This section gives the amounts of storage required for a sample configuration consisting of a single server on the Compaq (DIGITAL) OpenVMS processor with:

- The disk on the server holding the server and client product modules and the toolkit
- System default objects
- 20 local queues, each with 50 messages of 1 KB total length; half of the messages are persistent messages
- A single client
- No message channels for distributed queuing

The storage requirements for the server in this configuration are:

- RAM: 16 MB for VAX, 32 MB for AXP
- Disk space for MQSeries for Compaq (DIGITAL) OpenVMS:
  - Product modules: 16 MB for VAX, 18 MB for AXP
  - Paging space: 64 MB
  - Message queues: 1.5 MB
  - Logs: 12 MB (circular log)

The storage requirements for the clients in this configuration are small compared to the storage required for the operating system on the platforms.

Capacity planning figures for MQSeries

The following table shows the notional limits of the number of operations that can be performed on each MQSeries-connected workstation. It is recommended that, for adequate performance, you do not exceed these limits.

The numbers quoted were measured on an Alpha Server 1000 4/266 server that had 128 MB of installed memory.

Table 13. MQSeries for Compaq (DIGITAL) OpenVMS, capacity planning

Resource	Maximum number
Queue managers (heavily used)	4
Connections (including channels)	112
Open queues	1024
Active channels	112
Clients	112
Note: The number of connections, active channels, and clients can be doubled if you use the MQSeries fastpath bindings.	

Storage planning

Part 4. Planning for MQSeries for OS/2 Warp

Chapter 19. Introduction to MQSeries for OS/2 Warp

The MQSeries for OS/2 Warp product is in two parts, the *server* and the *client*. The server runs on a machine that is capable of running OS/2 Warp; the clients provided with the product are for AIX, DOS, HP-UX, OS/2 Warp, Sun Solaris, Windows NT, Windows 3.1, Windows 95, and Windows 98.

MQSeries provides the MQI programming interface for use by application programs that are running on the server or the client processor. More detail on clients and servers is given in "Chapter 7. Introduction to MQSeries clients and servers" on page 49.

This chapter includes basic information about:
- "Planning for MQSeries"
- "Migration from previous versions of MQSeries" on page 103

For more information on the hardware and software environments needed by MQSeries, see "MQSeries for OS/2 Warp" on page 239.

Planning for MQSeries

This chapter helps you to plan for the introduction of MQSeries into your enterprise, and introduces the items that you need to consider when doing this planning.

There are several stages in planning for the use of MQSeries that you must go through. They are:
1. Preparing your applications for the use of MQSeries
2. Planning to include MQSeries in a network
3. Preparing to install MQSeries
4. Planning to set up MQSeries

A prime requirement for a message delivery system is that it must be reliable. Many functions are built into MQSeries to ensure that messages are not lost despite system failures, and are not delivered more than once. MQSeries uses logging and other facilities to support these functions.

You must also plan for the operation and administration of MQSeries in your enterprise and consider the implementation of an appropriate set of security facilities. Brief outlines of these planning operations are included in this chapter.

Preparing your applications for the use of MQSeries

MQSeries brings the Message Queue Interface (MQI) to your applications. This interface allows you to modify existing applications and to write new applications. The MQI removes much of the need to understand the network and communication systems that you use. Thus you can expect to generate applications more speedily than before. However, you must plan to take advantage of the MQI by planning its use in your applications and by understanding the ways in which it assists you.

You can find more information on how to use the MQI in your applications by referring to the *MQSeries Application Programming Guide*.

Interfacing with CICS with MQSeries

With MQSeries you can create application programs for the CICS for OS/2 or CICS for Windows NT transaction environment. These applications can use the MQI to communicate with CICS or non-CICS programs in any of the environments supported by the MQSeries products.

Planning to use MQSeries in a network

MQSeries uses the distributed queuing facility to exchange messages between MQSeries platforms, using either the SNA LU 6.2, TCP, SPX, or NetBIOS transmission protocols.

You must consider how you will attach MQSeries to a network, and how you will define the channels that will be used to exchange messages. You can simplify your definitions for distributed queuing by using MQSeries clusters.

According to the way that you have set your systems up, security checks can be performed at various times. You can provide your own security services, using the exit facilities provided by MQSeries. MQSeries also provides exits relating to DCE security.

"MQSeries interoperability summary" on page 223 shows the links that are possible to other MQSeries products, and the transmission protocols that can be used.

For further information about distributed queuing, refer to the *MQSeries Intercommunication* and the *MQSeries Queue Manager Clusters* manuals.

Installing MQSeries for OS/2 Warp

MQSeries for OS/2 Warp is installed by using Software Installer/2 (SI/2), an IBM product. Although this is a separate licensed program, an SI/2 module is provided with your copy of MQSeries for OS/2 Warp that requires no additional license to be purchased. The supplied version of SI/2 is restricted in such a way that it can only be used for the installation of MQSeries for OS/2 Warp.

To prepare for the actual installation, you need to plan how much disk space you require in your OS/2 Warp system to accommodate MQSeries for OS/2 Warp. Assistance is given in "Chapter 23. Storage planning for MQSeries for OS/2 Warp" on page 113 to help you plan the amount of space required.

When you have installed MQSeries, you can run the supplied installation verification test. You can find more information to help you with the installation of MQSeries for OS/2 Warp in the *MQSeries for OS/2 Warp V5.1 Quick Beginnings* manual.

Setting up MQSeries

After installation, MQSeries needs to be set up, and customized for your own use. This ensures that the appropriate operating system facilities are made available to MQSeries, and that your MQSeries system is correctly initialized and ready to work with your applications.

MQSeries uses configuration files to hold the product configuration information used for logging, communications protocols and installable components. After installing the product, you can edit these files to tailor the operation of the product to meet the requirements of your installation.

In addition, you need to do the following:

- Define queues.
  - Consider your naming conventions for queues.
- Define trigger processes.
- Define remote links.
  - Define associated transmission queues.
  - Consider your naming conventions for remote queues.
  - Consider your naming conventions for channels.

Note: The characters within the names given to all MQSeries objects are case sensitive. Therefore, be very careful when defining the names of objects, to select the appropriate uppercase or lowercase characters.

You can find more information on the setting up and customizing processes for MQSeries in the *MQSeries System Administration* manual.

Planning recovery services for MQSeries

MQSeries provides logging services to allow backup and recovery of the messaging system. “Chapter 20. Backup and recovery planning for MQSeries for OS/2 Warp” on page 105 introduces you to these facilities, and to the items that you need to consider in order to include MQSeries in your backup and recovery plans.

You can find more information on the backup and recovery facilities provided by MQSeries in the *MQSeries System Administration* manual.

Planning data security for MQSeries

MQSeries provides a number of security facilities for use by your applications. An introduction to these facilities is given in “Chapter 21. Security planning for MQSeries for OS/2 Warp” on page 109.

Administration of MQSeries

A summary of the administration facilities provided by MQSeries is given in the *MQSeries System Administration* manual.

Migration from previous versions of MQSeries

When you have migrated from MQSeries Version 2 to MQSeries Version 5 you will **not** be able to revert to Version 2. You should back up your system before installing the new version. This will enable you to back off the upgrade if necessary. If you do this however, you will not be able to recover the work performed by MQSeries Version 5.

With MQSeries Version 5, the system default objects are created automatically when you use the crtmqm command to create a queue manager. The sample MQSC definition file, AMQSCOMA.TST, is no longer provided. If you have used AMQSCOMA.TST to customize your settings for MQSeries Version 2, and you want to use the same settings with Version 5, save your version of the file before you install MQSeries Version 5. You can then use this file to create the Version 2 default objects. Alternatively, you can generate a new MQSC definition file if required.

A list of the system default objects for MQSeries Version 5 is provided in the *MQSeries System Administration* manual.

Euro symbol support

Support for the new euro currency symbol has been added to MQSeries. If you need to modify your applications to use this symbol, ensure that they use one of the coded character sets that include it. These are described in the *MQSeries Application Programming Reference* manual. If you need to change the coded character set used by your queue manager, use the CCSID attribute of the ALTER QMGR command (or the equivalent PCF command).

Chapter 20. Backup and recovery planning for MQSeries for OS/2 Warp

This chapter describes the background concepts of recovery and restart. It contains the following sections:

More details of the logging and recovery facilities are given in the *MQSeries System Administration* manual.

Logging with MQSeries

The basic premise of a messaging system is that messages entered into the system are assured of delivery to the destination. One of the ways of ensuring that messages are not lost, is to maintain a record of the activities of the queue manager that handles the receipt, transmission, and delivery of messages.

MQSeries does this by recording all the significant changes to the data controlled or managed by the queue manager in a log. This process is called *logging*. The data changes that are logged include the puts and gets of persistent messages to and from queues, changes to queue attributes, and channel activities.

The purpose of logging is to create and maintain a log that:

- Keeps records of queue manager changes
- Keeps records of queue updates for use by the restart process
- Is a source for restoration of data should there be a hardware or software failure

Each MQSeries log consists of a log control file, together with one or more log files for the storage of data.

The log control file is, as its name implies, used to control and monitor the use of the log files. It contains information relating to the size, location, next available file, and other data related particularly to the log files themselves. All the log files within one log are the same size; there is a default value for this size, but you can override this value when you set up the log.

MQSeries provides a log dump facility (DMPMQLOG). This enables you to format and display the contents of the log when doing problem determination.

Types of logging with MQSeries

In MQSeries, there are two approaches to maintaining records of queue manager activities:

- Circular logging
- Linear logging

Each type of logging stores the recorded data in a set of files. The differences between the two types of logging are the contents, and the way that the files are linked together.

With circular logging, the set of log files are effectively linked together so as to form a ring. When data is collected, it is written sequentially into the files in such a way as to reuse the log files in the ring. You can use circular logging for:

- Crash recovery - that is, after a system failure of some kind has stopped the queue manager unexpectedly
- Restart recovery - after a planned close down of the system

With linear logging, the log is maintained as a continuous sequence of files. When data is collected, it is written sequentially into the log files; the space in the files is not reused, so that you can always retrieve any record from the time that the queue manager was created.

Because disk space is finite, you might have to plan for some form of archiving. Also, if you are handling a high volume of persistent messages, all your log files will eventually be filled. This will result in operator messages being written to an error log file, and some action will need to be taken by the system administrator to make more log space available, or to reuse the existing space. You can use linear logging for:

- Crash recovery
- Restart recovery
- Media recovery - to recreate lost or damaged data after a media failure by replaying the contents of the log

Selecting a logging method with MQSeries

You must base your selection of log type on your requirements for recovery.

Both types of logging can cope with unexpected power outages in the absence of hardware failure. If you accept that only crash or restart recovery is required, circular logging might be adequate. If media recovery is important to you, select linear logging.

With each type of logging, you will need to decide on the number of files to use in the log, and their size. The total amount of space needed will depend on the amount of data to be recorded, which depends on various parameters, including:

- The size of messages
- The number of gets and puts to queues
- The number of messages being transmitted by the message channel agents

Resource management with MQSeries

MQSeries supports the coordination of transactions by an external transaction manager that uses the X/Open XA interface. MQSeries can also act as a transaction manager, coordinating updates made by external resource managers.

MQSeries as a resource manager

MQSeries supports the coordination of transactions by an external transaction manager that uses the X/Open XA interface.

In an XA configuration, the MQSeries queue manager acts as an XA resource manager, managing message queues. The XA transaction manager coordinates the operations of the queue manager, and any other resource managers, to synchronize the commit or backout of transactions. This ensures that updates to MQSeries message queues are coordinated with the updates to all the other types of resource being managed.

CICS for OS/2 Warp does not operate as an XA transaction manager, so the XA resource management capabilities of MQSeries for OS/2 Warp and cannot be used. However, CICS does provide a single-phase commit process (as opposed to the two-phase XA coordination), which MQSeries takes part in.

This support is available only on the MQSeries server, and is not available to client applications. See the information on CICS transactions in the *MQSeries System Administration* manual for more details.

MQSeries as a transaction manager

MQSeries can act as a transaction manager and coordinate updates made by external resource managers within MQSeries units of work. These external resource managers must comply to the X/Open XA interface. MQSeries can act as a transaction manager for DB2.

Recovering from problems with MQSeries

MQSeries will recover from both communications failures and power loss incidents. In addition, it is sometimes possible to recover from other types of problem with the MQSeries data, such as inadvertent deletion of a file.

In the case of a communications failure, messages simply remain on queues until they are removed from the queues by a receiving application. If the message is being transmitted, it remains on the transmission queue until it can be successfully transmitted. To recover from a communications failure, it will normally be sufficient simply to restart the channels using the link that failed.

On a restart after your system has lost power, the queue manager will restore all the persistent messages on the queues to the state that existed just before the power failure, so that no persistent messages are lost; nonpersistent messages are discarded.

There are ways in which an MQSeries object can become unusable, for example due to inadvertent damage. In such situations, you will have to take steps to recover either your complete system or some part of it. The action required depends on when the damage is detected, whether the log method selected supports media recovery, and which object or objects are damaged.

Chapter 21. Security planning for MQSeries for OS/2 Warp

In general, each MQSeries product provides security facilities by building on those provided by the platform for which the product was designed.

This chapter describes the security facilities provided by MQSeries. The chapter contains these sections:

- "Setting user IDs with MQSeries"
- "Security exits for MQSeries"

Setting user IDs with MQSeries

MQSeries for OS/2 Warp provides the *User ID* installable component. This can be used to generate a user ID value, which the queue manager places in the message context portion of the message descriptor of all messages that are generated by application programs running on the MQSeries for OS/2 Warp client or server.

This user ID can be used by the programs that receive the messages, to verify that the messages have come from an authorized user.

Details of the User ID installable component can be found in the *MQSeries Programmable System Management* manual.

Security exits for MQSeries

The message channels that are used for distributed queuing, and the MQI channels that are used between clients and servers, both have security exit facilities that can invoke programs you have supplied.

For more information on these security exit programs, see the *MQSeries Intercommunication* manual.

Chapter 22. Administration of MQSeries for OS/2 Warp

This chapter is a summary of the administration facilities provided by MQSeries. It has the following sections:

- "Managing objects with MQSeries"
- "Remote administration with MQSeries" on page 112

Details of the commands, command interfaces, and utilities that are provided by MQSeries are given in the *MQSeries System Administration* manual.

Managing objects with MQSeries

It is the administrator's job to monitor MQSeries and make any changes that might be necessary. To do this, the administrator needs to know where each MQSeries object resides, what its characteristics are, and who has access to it.

The administrator can manage and monitor the resources using MQSeries commands (MQSC), or, if there are sets of commands that are issued regularly, by writing an application program that places them on the command queue.

Commands with MQSeries

MQSeries supports the following administration commands and facilities:

- MQSeries provides control commands that you can enter through the OS/2 Warp command line.
- You can use the **runmqsc** control command to cause MQSC commands from standard input to be executed.
- Any local or remote MQSeries application program can generate PCF commands in messages, and put them to the command queue, SYSTEM.ADMIN.COMMAND.QUEUE, to be processed by the MQSeries command server.

More information on how to use all these facilities is given in the *MQSeries System Administration* manual.

Changes made to the resource definitions of a queue manager using the commands (directly or indirectly) are preserved across restarts of the MQSeries system.

Managing communications with MQSeries

Part of the administrator's role is to ensure that the required communications links are activated, and to monitor the status of these links as required by your enterprise. You can find information describing these tasks in the *MQSeries Intercommunication* manual.

Remote administration with MQSeries

There are two aspects to the MQSeries remote administration facilities:

- MQSeries for OS/2 Warp can be used to manage remote systems.
- Other remote products can be used to manage an MQSeries for OS/2 Warp system.

Managing remote systems from MQSeries

Facilities are provided by MQSeries to allow an administrator to manage the following remote systems:

- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/390
- MQSeries for OS/2 Warp
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Because of the differences between the MQSeries products, it is not always possible to manage remotely the same set of MQSeries objects or attributes that you can manage locally.

If you wish to manage any other MQSeries product, you can write an application program to send the appropriate commands to the command queue for that product. However, some MQSeries products do not have a command queue, so they cannot accept commands from local or remote application programs.

Managing MQSeries from remote systems

MQSeries for OS/2 Warp can be managed from a remote MQSeries system, by an administrator using the facilities provided by the following products:

- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/390
- MQSeries for OS/2 Warp
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Note: You can manage MQSeries for OS/2 Warp from MQSeries for OS/390 by writing an application program to send the appropriate PCF commands to the command queue.

Chapter 23. Storage planning for MQSeries for OS/2 Warp

This chapter tells you how to plan the type and amount of storage you require when you include MQSeries in your system. It has the following sections:

- "RAM considerations for MQSeries"
- "Disk space considerations for MQSeries"
- "Capacity planning and performance figures for MQSeries" on page 114

RAM considerations for MQSeries

The processor memory (RAM) is used by MQSeries in the execution of the product modules, and as a paging area for the messages that are being processed. Parts of the paging area are written to, and read from, disk as necessary.

The minimum amount of RAM required to run the MQSeries server on OS/2 is 24 MB; if more RAM is available, the performance of the message processing improves. Allow 0.5 MB for each MQSeries client connected.

The amount of RAM required on each client system for an MQSeries client is small compared to that required for the operating system on each of the platforms.

Disk space considerations for MQSeries

Disk space is used by MQSeries for the following:

- Product modules - client and server executable modules, the toolkit, and the online documentation
- Paging space - server only
- Message queues - server only
- Logs - server only

Product modules for MQSeries

The disk space that you require for the product modules depends on the options that you decide to install: the options are described in the *MQSeries for OS/2 Warp V5.1 Quick Beginnings* manual.

Space might be required for client and server executable program modules, the toolkit, and the online documentation. If all options are selected, 66.5 MB is required.

The product modules can be stored on a LAN server, as an alternative to them being stored on a disk attached to the MQSeries client or server. MQSeries loads the modules from the LAN server when required.

Message queues for MQSeries

In order to estimate the total amount of storage that you will need for queues, you need to know:

- The number of queues that you have.
- The maximum number of messages there will be on each of the queues at any one time.
- The average size of message on each of the queues. The amount of storage required for one message varies. It is based on the size of the message data plus the size of the message header (456 bytes), rounded up the nearest 512-byte block. If you are using distribution lists, or grouped or segmented messages, the size of the header increases for the transmission queue.

Given these values, you can calculate the total amount of space required for queues. However, this value is likely to be an approximate value only, and it is advisable to add a contingency value, to avoid the situation where there is no space left for messages on the queues when your application is running.

The maximum size for a single queue on MQSeries for OS/2 Warp is 2 GB.

Log files for MQSeries

Significant events and data changes can be logged in circular or sequential logs. In particular, the logs are used for recording persistent messages.

All the log files in a log are of the same size. By default, this size is 4 MB, but this value can be changed by the system administrator when the log is defined.

For a circular log, the system administrator needs to specify how many files should be included in the log. For a sequential log, the number of files will increase over time, until the system administrator archives some of the files, and disposes of them. You need to plan for the permanent storage (diskettes, tape, or other media supported in your enterprise) that is to be used for these archived files.

Capacity planning and performance figures for MQSeries

Information about MQSeries performance and capacity planning is available on the Internet at:

`http://www.ibm.com/software/ts/mqseries/txppacs/txpm1.html`

Part 5. Planning for MQSeries for OS/390

Chapter 24. Introduction to MQSeries for OS/390

MQSeries for OS/390 runs on a System/390® processor that is capable of running OS/390, to provide messaging and queuing support for the following types of application:
- CICS
- IMS
- TSO and batch

This chapter provides basic information about:
- "Planning for MQSeries"
- "Installing and customizing MQSeries" on page 120
- "Migrating from MQSeries for MVS/ESA" on page 121

"Chapter 29. Storage planning for MQSeries for OS/390" on page 141 gives information on the storage requirements for MQSeries for OS/390. For more information on the hardware and software requirements for MQSeries for OS/390, see "MQSeries for OS/390" on page 241.

Planning for MQSeries

This chapter helps you to plan for the introduction of MQSeries for OS/390 into your enterprise. "Chapter 1. Introduction to IBM MQSeries" on page 3 introduces you to the concepts you must consider.

This chapter does not provide you with detailed information to enable you to perform the installation of MQSeries for OS/390. That information is provided in the *MQSeries for OS/390 Program Directory*. Equally, you are directed to other publications for detailed information on particular topics.

There are several stages in planning for the use of MQSeries for OS/390 that you must go through. They are:
1. Preparing your applications for the use of MQSeries for OS/390
2. Planning to include MQSeries for OS/390 in a network
3. Preparing to install MQSeries for OS/390
4. Planning to customize MQSeries for OS/390

A prime requirement for a message delivery system is that it must be reliable. Many functions are built into MQSeries for OS/390 to ensure that:
- Messages are not lost despite system failures.
- Messages are not delivered more than once.
- Messages are not accessed by, or delivered to, unauthorized persons or applications.

The features provided by MQSeries for OS/390 to support this, and which you must plan for, include:

- Logging (including options for dual logging and archiving)
- Automatic recovery from transaction, system, and storage media failures (for which suitable backups are needed)
- Restart from backup files
- Access to security management facilities

You must also plan for the administration of MQSeries for OS/390 in your operation, make decisions regarding the performance aspects of the product, and consider, if necessary, possible migration of both your system and applications from other products. Brief outlines of these planning operations are included in this chapter.

Preparing your applications for the use of MQSeries

MQSeries for OS/390 brings the Message Queue Interface (MQI) to your applications. This interface allows you to modify existing applications and to write new applications. The MQI removes much of the need to understand any network or communication systems that you use. Thus you can expect to complete applications more speedily than before. However, you must plan to take advantage of the MQI by planning its use in your applications and by understanding the ways in which it assists you.

You can find more information on how to use MQSeries for OS/390 in your applications by referring to the *MQSeries Application Programming Guide*.

Interfacing with CICS, IMS, or Batch

With MQSeries for OS/390 you can create applications in these environments:

- A CICS transaction environment
- An IMS transaction environment
- An OS/390 Batch and Time Sharing Option (TSO) environment, optionally using RRS

Applications (or transactions) connect to MQSeries by means of an *adapter*. There are adapters for each of these environments included in MQSeries for OS/390.

The MQSeries-IMS bridge

The MQSeries-IMS bridge is a component of MQSeries for OS/390 that allows direct access from MQSeries applications to applications on your IMS system. It enables implicit MQSeries API support. This means that you can re-engineer legacy applications that were controlled by 3270-connected terminals to be controlled by MQSeries messages, without having to rewrite, recompile, or relink them.

The bridge is an IMS *Open Transaction Manager Access* (OTMA) client.

The MQSeries-CICS bridge

The MQSeries-CICS bridge enables an application, not running in a CICS environment, to run a *program* or *transaction* on CICS and get a response back. This non-CICS application can be run from any environment that has access to an MQSeries network that encompasses MQSeries for OS/390.

A *program* is a CICS program that can be invoked using the EXEC CICS LINK command. It must conform to the DPL subset of the CICS API, that is, it must not use CICS terminal or syncpoint facilities. A *transaction* is a CICS transaction

designed to run on a 3270 terminal. This transaction can use BMS or TC commands. It can be conversational or part of a pseudoconversation. It is permitted to issue syncpoints.

Planning to use MQSeries in a network

MQSeries for OS/390 uses the distributed queue management (DQM) facility to exchange messages between MQSeries platforms.

With MQSeries for OS/390, access to remote queues can be with:
- SNA LU 6.2
- TCP
- CICS (using ISC)

Which of these alternative methods of connection is best for your installation depends on a number of factors, including:

- Whether you require CICS for use by your applications
- What other MQSeries products you wish to connect to, and the methods of connection that they support
- If there are alternatives possible, whether you require the best possible performance

You need to decide which of these protocols you are going to use before you install MQSeries for OS/390. "MQSeries interoperability summary" on page 223 shows the connections that are possible with networks containing MQSeries products.

Having decided on which form of connection to use, you need to define the channels that are to be used in the exchange of messages. You can simplify your definitions for distributed queuing by using MQSeries clusters (not available if you are using CICS ISC for distributed queuing).

For further information about distributed queuing, refer to the *MQSeries Intercommunication* and the *MQSeries Queue Manager Clusters* manuals. Refer to the *CICS Intercommunication Guide* for details on managing remote CICS links.

Using the OS/390 Workload Manager with MQSeries

If you want to use the queuing services of the OS/390 workload manager (WLM) with MQSeries, you need to use MQSeries *Workflow*. Use the WLM to define different classes of service for your MQSeries messages. When you have done this, if you put messages on a special queue called a *WLM-managed queue*, Workflow will pass information taken from the message descriptor and the work information header to the WLM, and so determine which class of service to use for the message.

For a general explanation of the basic concepts of MQSeries Workflow, see the *MQSeries Workflow: Concepts and Architecture* manual (GH12-6285), and for information about planning to use it on OS/390, see the *MQSeries Workflow for OS/390: Customization and Administration* manual (SC33-7030). For information about the Workload Manager, see the *MVS Planning: Workload Management* manual (GC28-1761).

Planning recovery services with MQSeries

"Chapter 26. Backup and recovery planning for MQSeries for OS/390" on page 127 introduces the concepts of recovery management in MQSeries for OS/390

subsystems. This chapter leads you into the planning you must do to ensure that your data can be recovered in cases of network or system failures.

Planning data security with MQSeries

MQSeries for OS/390 does not provide any security facilities of its own, but uses those provided by existing security management facilities. This book explores the possible options you must select from to protect your data and users from unauthorized access.

Administration with MQSeries

A summary of the administration facilities provided by the MQSeries for OS/390 products is given in "Chapter 28. Administration of MQSeries for OS/390" on page 137. Full details of these facilities can be found in the *MQSeries Command Reference*, and in the *MQSeries for OS/390 System Management Guide*.

Installing and customizing MQSeries

MQSeries for OS/390 uses the standard OS/390 installation procedure. This section reminds you of that procedure and also introduces some of the work that must be planned in order to customize MQSeries for OS/390 to your enterprise's particular needs.

Before you install MQSeries for OS/390, you must decide the following:

- Which communications protocol you are going to use
- Whether you are going to install one of the following optional national language features:
  - Japanese
  - Simplified Chinese
  - U.S. English (uppercase)
- Whether you are going to install the optional Client Attachment feature

You also need to plan how much storage you require in your OS/390 system to accommodate MQSeries for OS/390. Assistance is given in "Chapter 29. Storage planning for MQSeries for OS/390" on page 141 to help you plan the amount of storage required for MQSeries for OS/390. The chapter also gives an indication of the amount of storage you should plan to use, for example, for logs, page data sets, and objects.

Installing MQSeries

MQSeries for OS/390 is supplied with a Program Directory that contains specific instructions for installing the program on an OS/390 system. You must follow the instructions in the *MQSeries for OS/390 Program Directory*. They include not only details of the installation process, but also information about the prerequisite products and their service or maintenance levels.

SMP/E, used for installation on the OS/390 platform, validates the service levels and prerequisite and corequisite products, and maintains the SMP/E history records to record the installation of MQSeries for OS/390. It loads the MQSeries for OS/390 libraries and checks that the loads have been successful. You then have to customize the product to your own requirements.

Customizing MQSeries

MQSeries 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. Customization is described in detail in the *MQSeries for OS/390 System Management Guide*. However, below are the items you must consider when planning to customize your system.

You need to:

- Choose which language you want to use.
- Choose the distributed queuing facility.
- Define the MQSeries subsystem to OS/390.
- Update the OS/390 link list.
- APF authorize the MQSeries load libraries.
- Update the OS/390 program properties table.
- Create procedures for the MQSeries subsystem.
- Create the channel initiator procedures.
- Implement your ESM security controls.
- Customize the initialization input data sets.
- Create the bootstrap and log data sets.
- Define your page sets.
- Tailor your system parameter modules.
- Tailor the channel initiator parameter module.
- Set up the Batch/TSO and RRS adapters.
- Set up the operations and control panels.
- Set up the CICS and IMS adapters.
- Suppress information messages.
- Set up the CICS and IMS bridges.

Verifying your installation of MQSeries

After the installation and customization has been completed, you can use an installation verification program (IVP) supplied with the product to verify that the installation has been completed successfully. The IVP supplied is an assembler language program and should be run after MQSeries for OS/390 has been customized to suit your enterprise's needs. Details of the IVP and customization are given in the *MQSeries for OS/390 System Management Guide*.

Migrating from MQSeries for MVS/ESA

If you are migrating from MQSeries for MVS/ESA you do not have to perform most of the customization tasks listed in "Customizing MQSeries". Information about what you need to do when migrating from a previous version is given in the *MQSeries for OS/390 System Management Guide*.

When you have migrated to MQSeries for OS/390 Version 2.1, you will **not** be able to revert to previous versions of MQSeries for MVS/ESA.

Chapter 25. Data sets used by MQSeries for OS/390

This chapter provides a general introduction to the specialized data sets used by MQSeries for OS/390. MQSeries for OS/390 uses data sets called *page sets* for storing messages. It maintains *logs* of data changes and significant events as they occur to provide backup and recovery facilities. The logs contain information about queues and messages. The *bootstrap data set (BSDS)* stores information about the data sets that contain the logs.

This chapter contains basic information about:

Page sets for MQSeries

A page set is a linear VSAM data set that has been formatted for use by MQSeries for OS/390. Page sets are used primarily to store messages and object definitions. Each page set is identified by a page set ID (PSID), an integer in the range 00 through 99. In particular, MQSeries for OS/390 uses page set 00 to store queue definitions and other important information relevant to the queue manager.

Management of page sets is done through the use of buffer pools and storage classes.

Buffer pools and buffers for MQSeries

For efficiency, MQSeries for OS/390 implements a form of caching whereby messages are stored temporarily in buffers before being stored in page sets in DASD.

The buffers are organized into buffer pools. You can define up to four buffer pools for each MQSeries subsystem. Each buffer is 4 KB long. The maximum number of buffers is determined by the amount of storage available in the MQSeries address space. You are recommended to use four buffer pools.

Storage classes for MQSeries

A storage class maps one or more queues to a page set. When you define a queue you can also specify its storage class. More than one queue can use the same storage class, and you can define as many storage classes as you wish.

Log data sets for MQSeries

MQSeries for OS/390 records all *persistent* messages in the *active log* as they are put onto queues by applications. The active log resides on DASD. Entries to it are placed into a log buffer in main storage. The in-storage log buffers are written to an active log data set when any of the following occur:

- The log buffers become full.
- The write threshold is reached (as specified in the CSQ6LOGP macro).
- Certain significant events occur, such as a commit point.

The log contains the information needed to recover messages, queues, and the queue manager; it does not contain information on statistics, tracing, or performance evaluation. (The *MQSeries for OS/390 System Management Guide* shows how to specify destinations for trace, statistic, and performance data.) The active log contains information from all the queues being used with MQSeries for OS/390.

When the active log is full, MQSeries for OS/390 switches to the next available log data set and, if archiving has been switched on during customization, copies the contents of this log to an *archive log*. The archive log can be a data set on a direct access storage device (DASD) or on magnetic tape. If there is a problem, MQSeries for OS/390 uses these log entries to restore the message queues.

The archive log consists of up to 1000 sequential data sets. Each data set can be cataloged using the Integrated Catalog Facility (ICF).

MQSeries for OS/390 allows you to have either *single logging* or *dual logging*. Dual logging is used to minimize the likelihood of problems during restart. When dual logging is in use, MQSeries for OS/390 records the same information into two data sets.

Single logging gives you between 2 and 53 active log data sets. Dual logging gives you between 4 and 106 active log data sets. If possible, when using dual logging, the log data sets should be on separate volumes. This reduces the risk of them both being lost if the volume is corrupted or destroyed. Each active log data set is a single-volume, single-extent VSAM linear data set.

There is no relationship between the number of queues and the number of log data sets that you have. Instead, the number of data sets required is dependent on the amount of message traffic that you plan for. An algorithm is given in “Logs and archive storage for MQSeries” on page 142 to help you to calculate the amount of storage you should reserve for your logs.

For a complete description of logs, their contents, customization, and archiving, see the *MQSeries for OS/390 System Management Guide.*

Bootstrap data set for MQSeries

The *bootstrap data set* (BSDS) is a VSAM key-sequenced data set (KSDS) that holds information needed by MQSeries for OS/390. It contains:

- An inventory of all active and archived log data sets known to MQSeries for OS/390. This inventory is used by MQSeries for OS/390 to:
  - Track the active and archived log data sets.
  - Locate log records so that it can satisfy log read requests during normal processing.
  - Locate log records so that it can handle restart processing.

  MQSeries for OS/390 stores information in the inventory each time an archive log data set is defined or an active log data set is reused. For active logs, the inventory shows which are full and which are available for reuse. The inventory holds the relative byte address (RBA) of each log data set.
- A *wrap-around* inventory of all recent MQSeries for OS/390 activity. This is needed if MQSeries for OS/390 has to be restarted.

The active logs are first registered in the BSDS when MQSeries for OS/390 is initiated. They cannot be replaced, nor can new ones be added, without terminating and restarting MQSeries. The BSDS is required if the subsystem has an error and has to be restarted. To minimize the likelihood of problems during a restart, MQSeries can be configured with dual BSDSs, each recording the same information. This is known as running in "dual mode". As for dual active log data sets, copies should, if possible, be on separate volumes. This reduces the risk of them both being lost if the volume is corrupted or destroyed.

Archive log data sets and BSDS copies for MQSeries

A copy of the BSDS is placed in each new archive data set. If the archive log is on tape, the BSDS copy is the first file on the first output volume. If the archive log is on DASD, the BSDS copy is a separate file on the same volume.

What a log contains for MQSeries

An active log can contain up to 2^{48} bytes. Each byte in the active log can be addressed by its offset from the beginning of the log, and that offset is known as its *relative byte address* (RBA).

The log is made up of *log records*, each of which is a set of log data treated as a single unit. A log record is identified by the RBA of the first byte of its header; that RBA is called the relative byte address *of the record*. The RBA is like a time stamp because it uniquely identifies a record that starts at a particular point in the log.

Each log record has a header that gives its type, the MQSeries component that caused the record to be made, and, for unit of recovery records, the unit of recovery identifier.

Most of the log records describe changes to MQSeries queues. All such changes are made within units of recovery.

Checkpoint records for MQSeries

To reduce restart time, MQSeries for OS/390 takes periodic checkpoints during normal operation:

- When a predefined, customized number of log records has been written
- At the end of a successful restart
- At normal termination

At a checkpoint, MQSeries logs its current status and registers the RBA of the checkpoint in the bootstrap data set (BSDS). At restart, MQSeries uses the information in the checkpoint records to reconstruct the state it was in when it terminated.

Many log records can be written for a single checkpoint.

Chapter 26. Backup and recovery planning for MQSeries for OS/390

Developing backup and recovery procedures at your site is vital to avoid costly and time-consuming losses of data. MQSeries for OS/390 provides the means for recovering both queues and messages to their current state after a system failure. You should develop procedures for backing up page sets and creating a point of consistency.

This chapter describes what you should consider to minimize problems following any system failure. It contains information about:

- "Planning your logging environment with MQSeries"
- "Planning your archive storage with MQSeries"
- "Other recovery considerations with MQSeries"
- "General tips for backup and recovery with MQSeries" on page 130

Planning your logging environment with MQSeries

The MQSeries for OS/390 logging environment is established during customization to specify options, such as whether to have single or dual active logs, what media to use for the archive log volumes, and how many log buffers to have.

In a production subsystem, it is important to establish dual archiving and dual logging to minimize the risk of losing your data (for example, because of DASD failures).

Planning your archive storage with MQSeries

This section describes the different ways of maintaining your archive log data sets.

Archive log data sets can be placed on standard-label tapes, or DASD, and can be managed by Data Facility Hierarchical Storage Manager (DFHSM).

Archive log data sets are dynamically allocated, with names chosen by MQSeries for OS/390. The data set name prefix, block size, unit name, and DASD sizes needed for such allocations are specified when MQSeries for OS/390 is customized. You can also choose, at customization time, to have MQSeries for OS/390 add a date and time to the archive log data set name. You can change this information by customizing MQSeries for OS/390 again.

If you specify dual archive logs at installation time, each log *control interval* (CI) retrieved from the active log is written to two archive log data sets. The log records that are contained on the pair of archive log data sets are identical, but the end-of-volume points are not synchronized for multivolume data sets.

Other recovery considerations with MQSeries

In addition to the logging and archiving facilities provided by MQSeries for OS/390, you should also consider other space management and recovery facilities that are provided in your enterprise. The following sections provide a reminder of some of these facilities.

CICS recovery

CICS recognizes MQSeries for OS/390 as a non-CICS resource (or external resource manager), and includes MQSeries for OS/390 as an agent in any syncpoint coordination requests using the CICS resource manager interface (RMI). For more information about CICS recovery, see the *CICS Recovery and Restart Guide*. For information about the CICS resource manager interface, see the *CICS Customization Guide*.

IMS recovery

IMS/ESA recognizes MQSeries for OS/390 as an external subsystem and as a participant in syncpoint coordination. IMS recovery for external subsystem resources is described in the *IMS/ESA Customization Guide*.

RRS recovery

MQSeries can participate in units of recovery managed by OS/390 Recoverable Resource Manager Services (RRS). This means that MQSeries batch applications can participate in two-phase commit processing with other products that support RRS (for example, DB2). For more information, see the *MQSeries Application Programming Guide*.

Backup and recovery with DFHSM

The data facility hierarchical storage manager (DFHSM) does automatic space and data availability management among storage devices in your system. If you use it, you need to know that it moves data to and from MQSeries for OS/390 storage automatically.

DFHSM manages your DASD space efficiently by moving data sets that have not been used recently to less expensive storage. It also makes your data available for recovery by automatically copying new or changed data sets to tape, or DASD, backup volumes. It can delete data sets, or move them to another device. Its operations occur daily, at a specified time, and allow for keeping a data set for a predetermined period before deleting or moving it.

All DFHSM operations can also be performed manually. The *Data Facility Hierarchical Storage Manager User's Guide* explains how to use the DFHSM commands.

If you use DFHSM with MQSeries for OS/390, be aware that DFHSM:
- Uses cataloged data sets
- Operates on page sets and logs

Using Extended Recovery Facility (XRF)

MQSeries for OS/390 can be used in an Extended Recovery Facility (XRF) environment. All MQSeries for OS/390-owned data sets (executable code, BSDSs, logs, and page sets) must be on DASD shared between the active and alternate XRF processors. If you use XRF for recovery, you must stop MQSeries for OS/390 on the active processor and start it on the alternate. For CICS, this can be done using the command list table (CLT) provided by CICS, or manually by the system operator. For IMS, this is a manual operation and must be done after the coordinating IMS system has completed the processor switch.

MQSeries for OS/390 utilities must be completed or terminated before MQSeries for OS/390 can be switched to the alternate processor. Consider the effect of this potential interruption carefully when planning your XRF recovery plans.

Take care to prevent MQSeries for OS/390 starting on the alternate processor before the MQSeries for OS/390 system on the active processor terminates. A premature start can cause severe integrity problems in data, the catalog, and the log. Using global resource serialization (GRS) helps avoid the integrity problems by preventing simultaneous use of MQSeries for OS/390 on the two systems. The BSDS must be included as a protected resource, and the active and alternate XRF processors must be included in the GRS ring.

Using the OS/390 Automatic Restart Manager (ARM)

MQSeries for OS/390 can be used in conjunction with the OS/390 automatic restart manager (ARM). If a queue manager or a channel initiator has failed, ARM will restart it on the same OS/390 image. If OS/390 fails, a whole group of related subsystems and applications will also fail. ARM can restart all the failed systems automatically, in a predefined order, on another OS/390 image within the sysplex. This is called a cross-system restart.

You can use ARM to restart an MQSeries subsystem that uses LU 6.2 communication protocols on a different OS/390 image within the sysplex in the event of OS/390 failure. (You cannot do this if you use TCP communication protocols.)

To enable automatic restart:

- You must set up an ARM coupling data set.
- You must define the automatic restart actions that you want OS/390 to perform in an *ARM policy*.
- The ARM policy must be started.
- The subsystem must register with ARM at startup.

If you want to restart queue managers in different OS/390 images automatically, every queue manager must be defined in each OS/390 image on which that queue manager might be restarted, with a sysplex-wide unique 4-character subsystem name.

Using MQSeries with ARM is described in the *MQSeries for OS/390 System Management Guide*.

Preparing for disaster recovery

In the case of a total loss of an MQSeries for OS/390 computing center, you can recover on another MQSeries for OS/390 system at a recovery site. To be able do this, you must regularly back up the data sets and logs and provide the means to transfer them to the recovery site if necessary.

General tips for backup and recovery with MQSeries

The MQSeries for OS/390 restart process recovers your data to a consistent state by applying log information to the page sets. If your page sets are damaged or unavailable, you can resolve the problem using your backup copies of your page sets (provided that all the logs are available). If your log data sets are damaged or unavailable, it might not be possible to recover completely. It is recommended that you do the following:

- Periodically take backup copies.
- Use dual logging for your active log, archive log, and bootstrap data sets.
- Keep archive logs you might need.
- Retain the DD name or page set association.

These are described in more detail below.

Periodically taking backup copies

A *point of consistency* is the term used to describe a set of backup copies of MQSeries for OS/390 page sets and the corresponding log data sets required to recover these page sets. These backup copies provide a potential restart point in the event of page set loss (for example, page set I/O error). If MQSeries for OS/390 were to be restarted using these backup copies, the data in MQSeries for OS/390 would be consistent up to the point that these copies were taken. Providing that all logs are available from this point, MQSeries for OS/390 can be recovered to the point of failure. See the *MQSeries for OS/390 System Management Guide* for more information about points of consistency.

The more recent your backup copies the quicker MQSeries for OS/390 can recover the data in the page sets. The recovery of the page sets is dependent on all the necessary log data sets being available.

In planning for recovery, you need to determine how often to take backup copies and how many complete backup cycles to keep. These values tell you how long you must keep your log data sets and backup copies of page sets for MQSeries for OS/390 recovery.

In deciding how often to take backup copies, consider the time needed to recover a page set. It is determined by:

- The amount of log to traverse
- The time it takes an operator to mount and remove archive tape volumes
- The time it takes to read the part of the log needed for recovery
- The time needed to reprocess changed pages

In general, the more often you make backup copies, the less time recovery takes; but, of course, the more time is spent making copies.

You should keep at least two copies of each cycle of page set backup. This reduces the risk involved if one backup copy is lost or damaged.

Using dual logging for your log data sets

This increases the chances of recovering from all problems. However, it also increases the processing overhead and will affect the performance of your system. You need to consider the gains and losses caused by dual logging.

If you use dual logging, keep the dual logs separate.

Keeping archive logs you might need

MQSeries for OS/390 can use archive logs during restart. You must keep sufficient archive logs so that the system can be fully restored. MQSeries for OS/390 might need to use an archive log to recover a page set from a restored backup copy. If you have discarded that archive log, MQSeries for OS/390 will not be able to restore the page set to its current state and you will have to resolve the problem manually. See the *MQSeries for OS/390 System Management Guide* for details of how recovery is effected using the archive log.

Retaining the DD name or page set association

MQSeries for OS/390 associates page set number 00 with DD name CSQP0000, page set number 01 with DD name CSQP0001, and so on up to CSQP0099. MQSeries for OS/390 writes recovery log records for a page set based on the DD name the page set is associated with. For this reason, you must not move or rename page sets that have already been associated with a page data set ID (PSID) DD name; otherwise, MQSeries for OS/390 tries to recover the page sets with the wrong log data.

Chapter 27. Security planning for MQSeries for OS/390

Because MQSeries for OS/390 handles information passing between subsystems, it needs the safeguard of a security system to ensure that the resources it owns and manages are protected and secure from unauthorized access. It needs to ensure that the following are not accessed or changed by any unauthorized person or process:

- Connections to the MQSeries for OS/390 subsystem
- Resources such as queues, processes, and namelists
- MQSeries links to remote queue managers
- MQSeries system control commands
- MQSeries messages
- Context information in messages

To provide the necessary security, MQSeries for OS/390 uses the OS/390 System Authorization Facility (SAF) to route authorization requests to an external security manager (ESM), such as the Resource Access Control Facility (RACF). MQSeries does no verification of its own.

This book assumes that you are using RACF as your ESM. If you are using a different ESM, you might need to modify the techniques mentioned in this book.

The decision to allow access to an object is made by the ESM. MQSeries follows the decision made by the ESM. If the ESM cannot make a decision, MQSeries does not allow access to the object.

If you are planning a distributed system, you must agree on the levels of security checking that are provided and used by the various administrative domains that exist across these distributed systems. Such agreement must also cover network security aspects and access by clients.

This chapter includes basic information about:

- "MQSeries security overview" on page 134
- "Security exits with MQSeries" on page 135
- "Things to consider when setting up security" on page 135

For further information about security facilities in MQSeries for OS/390, refer to the *MQSeries for OS/390 System Management Guide*.

MQSeries security overview

MQSeries for OS/390 provides you with a range of choices when specifying your security requirements. The choice ranges from no security checking to full checking.

Subsystem security with MQSeries

Subsystem security allows you to control whether any security checking is done on the whole subsystem. You can set security checking according to how secure you decide your users are in a particular subsystem. For example, you might apply full security checking on your production system, but have no checking on your test system.

If you decide you want subsystem security, you also need to consider the following:

- Connection security

  Do you want to control who and what connects to your MQSeries for OS/390 subsystem?

- API resource security

  Resources can be security checked when a user issues **MQOPEN** or **MQPUT1** calls to the API. The security checking of the API resources is subdivided into the following:

  – Queue security

    Queue security allows you to control who is allowed to open a queue and how they are allowed to access it.

  – Process security

    Process security allows you to control who is allowed to open a process.

  – Namelist security

    Namelist security allows you to control who is allowed to open a namelist.

  – Alternate user security

    Alternate user security is used to control whether a particular user can use another user ID's authority to open an object on that user ID's behalf.

  – Context security

    Context information consists of two parts, an identity section and an origin section. The identity section specifies who the message came from; the origin section specifies where the message came from and when it was put to the queue.

    Context security allows you to control who is allowed to use the context-related options.

- Command resource security

  Command resource security allows to you to control the resources a command can access. Command resource security is independent of API security. For example, user A might be allowed to define a local queue but might not be allowed to get messages from that queue.

- Reslevel security

  Reslevel security is not a type of security but an option that allows you to control the number of user IDs checked for MQSeries for OS/390 API resource security checking (where applicable).

- Command security

  Command security allows you to control who is allowed to issue an MQSeries for OS/390 command.

Security classes used by MQSeries

MQSeries for OS/390 uses its own special RACF classes to hold its security information. These special classes are described in the *MQSeries for OS/390 System Management Guide*.

Security exits with MQSeries

The message channels that are used for distributed queuing and the MQI channels that are used between clients and servers, have security exit facilities that can invoke programs supplied by you. Examples of the types of function for which these exit facilities are intended include:

- Verification that the partners at the ends of the channel are genuine, and have the appropriate security authorizations to take part in the exchange
- Encryption and decryption of messages

For more information about security exits, refer to the *MQSeries Intercommunication* manual.

Things to consider when setting up security

In order to help you set up your security you need to consider the following:

- Decide what levels of security you require (from above list).
- Decide what types of resources are going to be used and, preferably, set up a naming convention for them (for example, queues and their names).
- Decide who will be using the system and the resources you have. Consider which users and user IDs will use which resources and so on.
- Decide how they are going to be using the system and what access levels each user requires for each resource they use.
- Decide any grouping of users and resources that might be required.
- Decide which users will be accessing or connecting to MQSeries for OS/390 and how they will be doing so; for example, batch, CICS, IMS, and so on.
- Decide on how to manage network security, for example to ensure that all incoming messages are from authorized applications or users and that all outgoing messages are from authentic users and are secure.

For details on how the steps can be carried out to satisfy your security requirements, refer to the *MQSeries for OS/390 System Management Guide*.

Chapter 28. Administration of MQSeries for OS/390

This chapter is a summary of the administration facilities provided by MQSeries for OS/390. It has the following sections:

Details of the administration facilities provided by MQSeries for OS/390 are given in the *MQSeries Command Reference*, and in the *MQSeries for OS/390 System Management Guide*.

Managing objects with MQSeries

It is the administrator's job to monitor MQSeries for OS/390 and make any changes that might be necessary. To do this, the administrator needs to know where each MQSeries object resides, what its characteristics are, and who has access to it. The administrator can manage and monitor the resources by use of the commands, the operations and control panels, or the utility programs supplied with MQSeries for OS/390.

If there are sets of commands that are issued regularly on your system, you can run a supplied utility program, or write a program that constructs commands and places them on the command queue. Details of how to write these administration programs are contained in the *MQSeries for OS/390 System Management Guide*.

Commands on MQSeries

MQSeries for OS/390 supports MQSC commands, which can be issued from the following sources:

- The OS/390 console.
- The initialization input data set, CSQINP1, to be processed before the restart phase of MQSeries for OS/390 initialization.
- The initialization input data set, CSQINP2, to be processed after the restart phase of MQSeries for OS/390 initialization.
- The initialization input data set, CSQINPX, to be processed whenever the channel initiator is started.
- The supplied batch utility, CSQUTIL, processing a list of commands in a sequential data set.
- A suitably authorized application, by sending a command as a message to the command queue. This can be:
  - A batch region program
  - A CICS application
  - An IMS application
  - A TSO application
  - An application program or utility on another MQSeries system

Much of the functionality of these commands is available in a user-friendly way from the MQSeries for OS/390 operations and controls panels. Information on using commands, and on using the operations and controls panels is given in the *MQSeries for OS/390 System Management Guide*.

MQSeries for OS/390 can perform security checks to ensure that the user is authorized to issue particular commands for particular resources.

Changes made to the resource definitions of a queue manager using the commands (directly or indirectly) are preserved across restarts of the MQSeries subsystem.

MQSeries for OS/390 does not support the PCF commands.

Managing communications with MQSeries

Part of the administrator's role is to ensure that the required communications links are activated, and to monitor the status of these links as required by your enterprise. You can find information describing these tasks in the *MQSeries Intercommunication* manual.

Remote administration with MQSeries

There are two aspects to the MQSeries remote administration facilities:

- MQSeries for OS/390 can be used to manage remote systems.
- Other remote products can be used to manage an MQSeries for OS/390 system.

Managing remote systems from MQSeries

Your administrator can manage remote MQSeries for OS/390 systems using facilities provided by MQSeries for OS/390.

Because of the differences between the MQSeries products, it is not always possible to manage remotely the same set of objects or attributes that you can manage locally.

If you wish to manage any other MQSeries product, you can write an application program to send the appropriate commands to the queue manager command queue. However, some MQSeries products do not have a command queue, and they cannot accept commands from local or remote application programs.

Managing MQSeries from remote systems

MQSeries for OS/390 can be managed from a remote MQSeries system, by an administrator using the facilities provided by the following products:

- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/390
- MQSeries for OS/2 Warp
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Managing accounting information with MQSeries

MQSeries for OS/390 provides an accounting trace to enable you to collect information that you can use to charge your customers for their use of your MQSeries for OS/390 subsystem. This facility is described in the *MQSeries for OS/390 System Management Guide*.

Using the MQSeries utilities

MQSeries for OS/390 supplies a set of utility programs to help you perform various administrative tasks. These utilities:

- Perform backup, restoration, and reorganization tasks.
- Issue commands and process object definitions.
- Generate data-conversion exits.
- Modify the bootstrap data set.
- List information about the logs.
- Print the logs.

For more information about all these utilities, see the *MQSeries for OS/390 System Management Guide*.

The CSQUTIL utility

The CSQUTIL utility program is provided with MQSeries for OS/390 to help you perform backup, restoration, and reorganization tasks, and to issue commands and process object definitions. Through this utility program, you can invoke the following functions:

COMMAND
To issue any of the MQSC commands described in the *MQSeries Command Reference* manual, to record object definitions, and to make client-channel definition files.

COPY To read the contents of a named MQSeries for OS/390 message queue or the contents of all the queues of a named page set, and put them into a sequential file and retain the original queue.

COPYPAGE
To copy whole page sets to larger page data sets.

EMPTY
To delete the contents of a named MQSeries for OS/390 message queue or the contents of all the queues of a named page set, retaining the definitions of the queues.

FORMAT
To format MQSeries for OS/390 page sets.

LOAD
To restore the contents of a named MQSeries for OS/390 message queue or the contents of all the queues of a named page set from a sequential file created by the COPY function.

RESETPAGE
To copy whole page sets to other page set data sets and reset the log information in the copy.

SCOPY
To copy the contents of a queue to a data set while the queue manager is offline.

SDEFS
To produce a set of define commands for objects while the queue manager is offline.

The data conversion exit utility

The MQSeries for OS/390 data conversion exit utility (CSQUCVX) runs as a stand-alone utility to create data conversion exit routines.

The change log inventory utility

The MQSeries for OS/390 change log inventory utility program (CSQJU003) runs as a stand-alone utility to change the bootstrap data set (BSDS). The utility can be used to:

- Add or delete active or archive log data sets.
- Supply passwords for archive logs.

The print log map utility

The MQSeries for OS/390 print log map utility program (CSQJU004) runs as a stand-alone utility to list the following information:

- Log data set name and log RBA association for both copies of all active and archive log data sets. If dual logging is not active, there is only one copy of the data sets.
- Active log data sets available for new log data.
- Contents of the queue of checkpoint records in the bootstrap data set (BSDS).
- Contents of the archive log command history record.
- System and utility time stamps.

The log print utility

The log print utility program (CSQ1LOGP) is run as a stand-alone utility. You can run the utility specifying:

- A bootstrap data set (BSDS)
- Active logs (with no BSDS)
- Archive logs (with no BSDS)

Chapter 29. Storage planning for MQSeries for OS/390

This chapter tells you how to plan the type and amount of storage you require when you include MQSeries for OS/390 in your system. It contains information about:

MQSeries for OS/390 is an OS/390 subsystem running in its own address space. The virtual storage usage therefore falls into two categories: the storage used by the subsystem address space itself and the additional storage used by applications requesting MQSeries for OS/390 services.

MQSeries address space storage

Each MQSeries for OS/390 subsystem (including the channel initiator) has the following storage requirements:

CSA 4 KB
ECSA 5 MB

In addition, each concurrent MQSeries task requires about 1500 bytes of ECSA. When a task ends, this storage can be reused by other MQSeries tasks. MQSeries does not release the storage until the queue manager is shut down, so the maximum amount of ECSA required can be calculated by multiplying the maximum number of concurrent tasks by 1500 bytes.

Concurrent tasks consist of the following:

- The number of Batch, TSO or IMS regions that have connected to MQSeries, but not disconnected
- The number of CICS transactions that have issued an MQSeries request, but have not terminated

The trace table also resides in the ECSA; you should use the TRACTBL parameter of the CSQ6SYSP macro to determine the size of the resident trace table. This macro is described in the *MQSeries for OS/390 System Management Guide*.

Private region storage usage

Every channel uses the following private region virtual storage below the 16 MB line in the channel initiator address space:

- 1200 bytes if LE/370 APAR PQ03507 has been applied
- None if LE/370 APAR PQ06157 has also been applied
- 8 KB otherwise

Every channel uses approximately 160 KB of extended private region in the channel initiator address space. Storage is increased if messages larger than 32 KB are transmitted.

This increased storage is freed when:

- A sending or client channel requires less than half the current buffer size for 10 consecutive sends.
- A heartbeat is sent or received.

The storage is freed for re-use within the LE environment, but is not seen as free by the OS/390 virtual storage manager.

This means that the upper limit for the number of channels is dependent on message size and arrival patterns as well as individual user system limitations on extended private region size. The limit is likely to be approximately 9000 channels on many systems.

Logs and archive storage for MQSeries

Active log data sets record significant events and data changes. They are periodically off-loaded to the archive log. Consequently, the space requirements for your active log data sets depend on the volume of messages that your MQSeries for OS/390 handles and how often the active logs are off-loaded to your archive data sets. MQSeries for OS/390 provides optional support for dual logging; if you use this your log storage requirement will be doubled.

If you decide to place the archive data sets on direct access storage devices (DASD), you need to reserve enough space on the devices. Space should also be reserved for the bootstrap data sets (BSDS). The functions of these data sets are described in “Chapter 25. Data sets used by MQSeries for OS/390” on page 123. The above are all separate data sets and should, preferably, be allocated space on different volumes and strings to minimize DASD contention and problems caused by any defects on the physical devices.

As each change to the system is logged, the size of storage required can be estimated from the size and expected throughput of messages. You must add to this a small overhead for the header information in the data sets.

To arrive at the size of the log extents, an algorithm can be developed which depends on various factors including the message rate, the message size in bytes, and how often you want to switch the log.

Below is shown an approximate calculation for the number of records to specify in the cluster for the log data set.

```
Number of records = (a * log switch interval required in seconds) / 4096

 where a =  (Number of puts/sec * Average message size)+440
          + (Number of gets/sec * 72)
          + (Number of units of recovery started  * 100)
          + (Number of syncpoints per second * 196)
```

Storage for page data sets and messages

The amount of storage needed for page data sets depends on the sizes of the messages that your applications will exchange, on the numbers of these messages, and on the rate at which they are created or exchanged. You can calculate the amount of storage needed for page data sets using the algorithm given in the *MQSeries for OS/390 System Management Guide*.

Storage for bootstrap data sets (BSDS)

Each BSDS requires 500 KB. You must have one BSDS but, optionally, can choose to have dual BSDSs. Further information about choosing single or dual BSDSs is given in "Chapter 25. Data sets used by MQSeries for OS/390" on page 123.

Planning your library storage for MQSeries

You need to allocate storage for the product libraries. The exact figures depend on your configuration, but an estimate of the space required by the distribution libraries is 35 MB. The target libraries also require about 35 MB. Additionally, you require space for the SMP/E libraries.

You should refer to the program directory supplied with MQSeries for OS/390 for information about the required libraries and their sizes.

Further information for MQSeries

This chapter has shown some of the main considerations when planning the storage required for MQSeries for OS/390. For more information about the library sizes, refer to the *MQSeries for OS/390 Program Directory*. For more information about storage variables to be defined during customization, refer to the *MQSeries for OS/390 System Management Guide*.

Chapter 30. Performance of MQSeries for OS/390

This chapter introduces the performance considerations that you must review when planning to include MQSeries for OS/390 in your system. These items include:

Impact of logging for MQSeries

The performance of MQSeries for OS/390 is sensitive to logging activities. MQSeries for OS/390 provides and maintains data integrity by the use of logs to record messaging events. If no logging were to take place, the performance of the system would depend primarily on processor availability.

MQSeries logs and page data sets are used in the same way as is typical for database systems. That is, messages are written to the log data set and held in virtual storage buffer pools backed by page data sets that are updated asynchronously.

Logs generally have a greater effect on the performance of MQSeries than they do on the performance of a database system. This is because typical database systems have a high read-only to update ratio, whereas in MQSeries, most **MQPUT** and **MQGET** operations are the equivalent of the addition and deletion of database rows respectively (an update operation). This means that MQSeries logs are typically written to more frequently than the logs for a database system. However, nonpersistent (nonrecoverable) messages are not written to the log data set.

Significant logging takes place mainly when using persistent messages, and particularly when persistent messages are being moved between queue managers. Hence, performance is affected by the number, and size, of persistent messages in the system, and also by the frequency of syncpointing and DASD performance for the log data sets.

Figure 11 on page 146 shows the response overhead added by logging for various message sizes in a typical CICS region. You will see that the response time increases by about 15%.

To achieve the best performance:

- Put active logs on the fastest DASD (for example, use DASD fast write).
- Ensure that when an active log is in use it is the only data set on that volume.

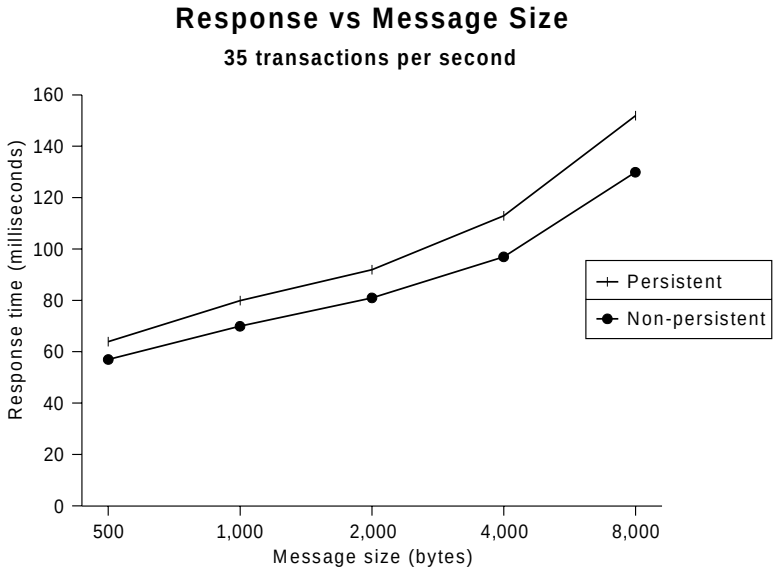

Figure 11. Impact of logging on response time

Impact of dual logging on MQSeries

MQSeries for OS/390 offers the capability of having dual logs. These logs should be placed on separate volumes. This ensures the log writes are not competing for disk accesses on the same volume and are therefore faster, as well as giving more protection against unexpected corruption of data on a volume.

Figure 12 on page 147 shows the overhead in terms of transaction response times in a CICS region with and without dual logging. Each transaction in Figure 12 on page 147 consists, on average, of five put or get messages, and covers a range of about 300 to 500 log events per second. The fast write facility is turned on in this example. For an explanation see "Fast write for logging on MQSeries" on page 147.

Ideally, data sets containing one copy of the log should be located on DASD accessed through a separate control unit and separate channel path from data sets containing the other copy.

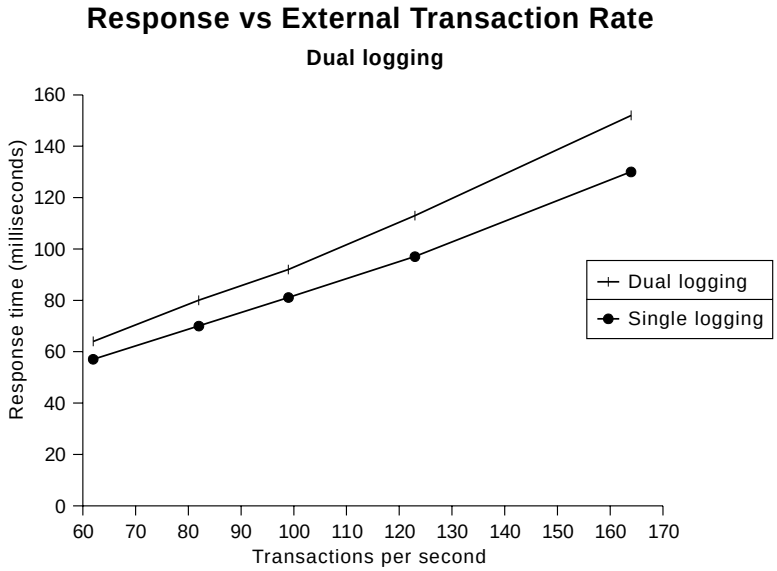

Figure 12. Impact of dual logging on response time

Fast write for logging on MQSeries

Fast write is a facility provided on some DASD controllers. It is a cache that is used to buffer write operations to DASD. Data can be written into the cache at channel speeds and is not held up by seek, rotational, and other delays in the DASD hardware.

Throughput gains can be made by directing the logs to a fast write facility if it is available. The facility significantly reduces I/O response times, as shown in Figure 13. The figure shows that if fast write is in use, a high throughput can be sustained beyond the point where, without fast write caching, the I/O time to complete would cause degradation in the transaction response time.

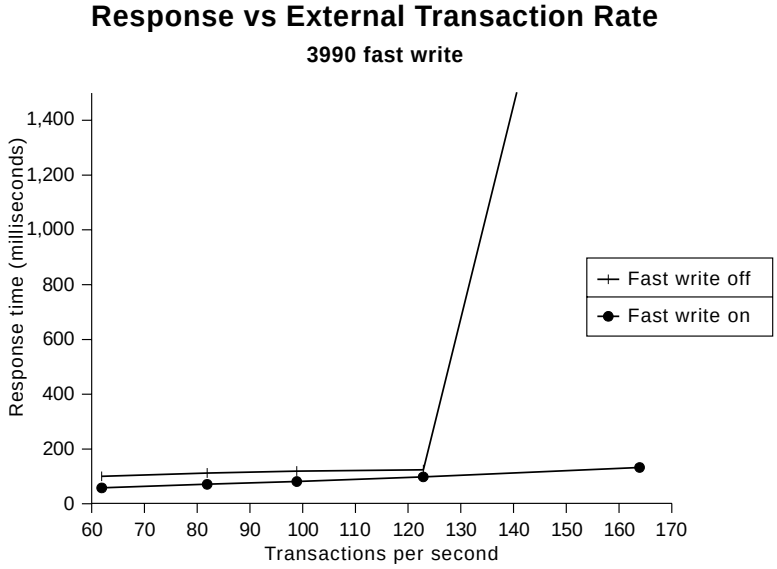

Figure 13. Impact of using 3990 fast write on response time

Causes of I/O to log for MQSeries

The active log buffer is written to DASD when a syncpoint is taken or when the threshold value set for flushing the buffer to DASD is reached. The log buffer is also written to DASD when a persistent message is placed on a queue and that message is not operating within a syncpoint.

Checkpointing on MQSeries

The purpose of a checkpoint is to harden committed messages to DASD periodically, and reduce the elapsed time required to recover messages in the event of recovery restart. The number of writes to the log buffer before a checkpoint is taken is defined when the system is customized. A checkpoint is also taken at each log switch.

At each checkpoint, all buffer pool pages that have changes not yet written to a page set, and older than the previous checkpoint, are written. Consequently, a higher number of log writes specified between checkpoints tends to decrease the amount of page set I/O, but increase the elapsed time for any recovery restart.

MQSeries page set I/O on MQSeries

Page set I/O is another activity that impacts performance. Hence, the more it can be avoided, the better the performance. You should specify enough buffer pool pages to contain the expected workload, so reducing excessive I/O. If the number you specify is insufficient, the I/O rate of the OS/390 system will increase.

At initialization you need to define the number of data buffers that are to be made available for use by queues. You can define between one and four buffer pools. Each buffer pool can hold between 100 and 125 000 buffers. You must define how many buffers you will need. Each buffer pool is also defined to a particular page data set ID (PSID).

You must also decide where the page data sets will be placed. When you have defined more than one PSID, it is best to locate the data sets on different physical volumes.

Buffer pools, page sets, storage classes, and queues for MQSeries

- Buffer pools are areas of MQSeries for OS/390 virtual storage reserved to satisfy the buffering requirements for one or more queues.
- Page sets are VSAM linear datasets and are each associated with a buffer pool.
- Storage classes are used to provide a mapping between queues and page sets.
- A queue is associated with a page set, which is a VSAM data set that is used when MQSeries for OS/390 moves data (for example, queues and messages) from buffers in main storage to permanent backing storage (DASD).

This gives a hierarchy which provides the following mappings:

- A queue (via its storage class) is associated with a single page set; many queues can map to the same page set.
- A page set is associated with a single buffer pool; many page sets can map to the same buffer pool.

In general, it is best to allocate as much storage as possible for the buffer pools. The larger the pool, the more likely that a message put to a queue will stay in storage long enough for it to be retrieved, thus saving I/O operations to the backing page set.

You should try to separate short-lived messages from messages that exist for a long time (for example half an hour). This means using different queues for the different messages, allocating the queues to different page sets, and allocating the different page sets to different buffer pools.

For short-lived messages you should allocate a large buffer pool. Long lived messages are written to the page set when the buffer pool fills up, or the messages have existed over two checkpoints. They are written to disk to reduce the startup time if the queue manager ends abnormally. Having a large buffer pool causes a lot of activity at a checkpoint, when the queue manager writes out the 'old' messages. Having a small buffer pool causes the data to be written to the page set at a steady rate, and reduces the impact at a checkpoint.

Refer to the *MQSeries for OS/390 System Management Guide* for more information about tuning your system.

Monitoring performance for MQSeries

A number of tools to monitor the performance of an MQSeries for OS/390 subsystem are provided. These include:
- Trace
- Display commands
- CICS adapter statistics

Further tools are provided by OS/390:
- System Management Facility (SMF)
- Generalized Trace Facility (GTF)

You can also use other IBM licensed programs:
- Service Level Reporter (SLR)
- Resource Management Facility (RMF™)
- CICS Monitoring facility

The use of these tools to monitor MQSeries for OS/390 performance is described in the *MQSeries for OS/390 System Management Guide*.

In addition to using these tools, with MQSeries for OS/390 you can write your own application programs, to use the event messages generated by the instrumentation events facility to produce performance reports.

More information on using instrumentation events is given in the *MQSeries Programmable System Management* manual.

Where to find more information

Information about MQSeries performance is available on the Internet at:

`http://www.ibm.com/software/ts/mqseries/txppacs/txpm1.html`

Chapter 31. Measured usage license charges with MQSeries for OS/390

Measured Usage License Charges (MULC) is a particular way of charging you for an IBM product that runs on an OS/390 system, based on how much use you make of the product. To determine the product usage, the OS/390 system records the amount of processor time that is used by the product when it executes.

OS/390 can measure how much processing time is spent in doing work on behalf of the MQSeries queue manager which is handling MQI calls, executing MQSeries commands, or performing some other action to support the messaging and queuing functions used by your application programs. The amount of processing time is recorded in a file at hourly intervals, and the hourly records are totalled at the end of a month. In this way, the total amount of time that has been used by the MQSeries for OS/390 product on your behalf is computed, and used to determine how much you should pay for your use of the MQSeries for OS/390 product that month.

MULC is implemented as follows:

- When MQSeries for OS/390 is installed, it identifies itself to OS/390, and requests that the *System Management Facilities (SMF)* mechanism within OS/390 is to automatically measure how much processor time is used by the MQSeries for OS/390 product.
- When enabled, the OS/390 usage measurement facility collects usage figures for each hour of the day, and generates usage records that are added to a report file on disk.
- At the end of one full month, these usage records are collected by a program, which generates a report of product usage for the month. This report is used to determine the charge for the MQSeries for OS/390 product.

More details on MULC can be found in *MVS/ESA Support for Measured License Charges*, GC28-1098.

Part 6. Planning for MQSeries for Tandem NSK

Chapter 32. Introduction to MQSeries for Tandem NSK

MQSeries for Tandem NSK runs on a machine that is capable of running the Tandem NSK operating system. MQSeries provides the MQI programming interface for use by application programs that are running on Tandem NSK.

For information on the hardware and software environments, see "MQSeries for Tandem NSK" on page 247.

Planning for MQSeries

This chapter helps you to plan for the introduction of MQSeries for Tandem NSK into your enterprise, and introduces the items that you need to consider when doing this planning.

There are several stages in planning for the use of MQSeries for Tandem NSK that you must go through. They are:

1. Preparing your applications for the use of MQSeries for Tandem NSK
2. Planning to include MQSeries for Tandem NSK in a network
3. Preparing to install MQSeries for Tandem NSK
4. Planning to set up MQSeries for Tandem NSK

A prime requirement for a message delivery system is that it must be reliable. Functions are built into MQSeries for Tandem NSK to ensure that:

- Messages are not lost despite system failures.
- Messages are not delivered more than once.
- Messages are not accessed by, or delivered to, unauthorized persons or applications.

MQSeries uses Tandem NSK facilities to support these functions.

You must also plan for the operation and administration of MQSeries for Tandem NSK in your enterprise, and consider the implementation of an appropriate set of security facilities. Brief outlines of these planning operations are included in this chapter.

Preparing your applications for use with MQSeries

MQSeries for Tandem NSK brings the Message Queue Interface (MQI) to your applications. This interface allows you to modify existing applications and to write new applications. The MQI removes much of the need to understand the network and communication systems that you use. Thus you can expect to generate applications more speedily than before. However, you must prepare to take advantage of the MQI by planning its use in your applications and by understanding the ways in which it assists you.

You can find more information on how to use the MQI in your applications by referring to the *MQSeries Application Programming Guide*.

Planning to use MQSeries in a network

MQSeries for Tandem NSK uses distributed queuing to exchange messages between MQSeries platforms, using either the SNA LU 6.2 or TCP transmission protocols.

You must consider how you will attach MQSeries for Tandem NSK to a network, and how you will define the message channels that will be used to exchange messages.

According to the way that you have set your systems up, security checks can be performed at various times. Various exits are provided that can be used by your applications to provide these facilities.

"MQSeries interoperability summary" on page 223 shows the links that are possible to other MQSeries products, and the transmission protocols that can be used.

For further information about distributed queuing, refer to the *MQSeries Intercommunication* manual.

Installing MQSeries

MQSeries for Tandem NSK is installed with the **instmqm** control command. For the D4x and G0x environments, a native version of MQSeries is available. For information about installing both native and nonnative versions of MQSeries for Tandem NSK, see the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Setting up MQSeries

After installation, MQSeries for Tandem NSK needs to be configured for use. This ensures that the appropriate Tandem NSK facilities are made available to MQSeries, and that your MQSeries system is correctly initialized and ready to work with your applications.

You need to do the following:

- Create a queue manager.
- Customize MQSeries for your Tandem system using PATHWAY to spread workload among CPUs.
- Define queues.
  - Consider your naming conventions for queues.
- Define trigger processes.
- Define remote links.
  - Define associated transmission queues.
  - Consider your naming conventions for remote queues.
  - Consider your naming conventions for channels.

Note: The characters within the names given to all MQSeries objects are case sensitive. Therefore, be very careful when defining the names of objects, to select the appropriate uppercase or lowercase characters.

You can find more information on the setting up and customizing processes for MQSeries for Tandem NSK in the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Planning recovery services with MQSeries

MQSeries for Tandem NSK provides facilities to allow backup and recovery of the messaging system. “Chapter 33. Backup and recovery planning for MQSeries for Tandem NSK” on page 159 introduces you to these facilities, and to the items that you need to consider in order to include MQSeries for Tandem NSK in your backup and recovery plans.

You can find more information on the backup and recovery facilities provided by MQSeries for Tandem NSK in the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Planning data security on MQSeries

MQSeries for Tandem NSK uses the facilities of the MQSeries *object authority manager (OAM)* installable component to control access to the various different types of queue manager resource (queues, process definitions, channels, and queue managers).

You can find more general information on authorization installable components in “Chapter 8. Introduction to the MQSeries Framework” on page 55.

“Chapter 34. Security planning for MQSeries for Tandem NSK” on page 161 introduces you to the security facilities provided by MQSeries for Tandem NSK and to some of the items you need to consider when planning for security.

You can find more information on the security facilities provided by MQSeries for Tandem NSK in the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Administration with MQSeries

Information about administering MQSeries is given in “MQSeries product administration facilities” on page 36.

A summary of the administration facilities provided for Tandem NSK is given in “Chapter 35. Administration of MQSeries for Tandem NSK” on page 163. Full details of these facilities can be found in the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Migration from MQSeries for Tandem NSK Version 1.5.1

If you are a user of MQSeries for Tandem NSK Version 1.5.1, you can convert your existing MQSeries configuration files and messages to work with MQSeries for Tandem NSK Version 2.2.0.1 using the following two conversion utilities:

CNV1520
Converts MQSeries for Tandem NSK Version 1.5.1 queue and channel definitions into MQSC scripts.

CNVMSGS
Transfers messages from MQSeries for Tandem NSK Version 1.5.1 message queues to Version 2.2.0.1 message queues after the queue definitions have been established using CNV1520.

Both utilities reside in the ZMQSEXE subvolume.

Migrating applications

To migrate your MQSeries for Tandem NSK Version 1.5.1 or Version 2.2 applications you must recompile and rebind them with Version 2.2.0.1 header files and libraries. Stubs have been provided for MQI calls that are not present or required in MQSeries for Tandem NSK Version 2.2.0.1, so code changes relating to MQSeries (other than including the correct header files) are not required. However, MQSeries for Tandem NSK Version 2 requires that you compile C programs with the WIDE model. MQSeries for Tandem NSK Version 1.5.1 required LARGE; if your programs contain code that relies on LARGE data representations, the code might have to be changed before it functions correctly under the WIDE model.

Chapter 33. Backup and recovery planning for MQSeries for Tandem NSK

This chapter describes the background concepts of recovery and restart. More information about the recovery facilities of MQSeries for Tandem NSK is given in the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Recovery facilities with MQSeries

MQSeries for Tandem NSK ensures that messages are not lost by using the Tandem NonStop Transaction Manager (TM/MP). TM/MP provides transaction protection, queue-file consistency, and queue-file recovery.

The TM/MP subsystem manages the complex operations for current transactions and database consistency, both user operations and MQSeries operations, making these operations transparent to both users and application programs.

A recovery restores the queue manager to the state it was in when the queue manager stopped. Any transactions that are in process are rolled back, removing from the queues any messages that were not committed at the time the queue manager stopped. Recovery restores all persistent messages; nonpersistent messages are lost during the process.

Recovering from problems with MQSeries

If you properly configure MQSeries and your NSK system software and hardware, the failure of any single hardware of software component does not result in the loss of any data or system functions. MQSeries can recover from a single point of failure while maintaining data integrity.

For more information about recovery from failures, see the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Backing up and restoring with MQSeries

Periodically, you might want to make a backup of your queue manager data to provide protection against possible corruption due to hardware failures. See the *MQSeries for Tandem NonStop Kernel System Management Guide* for information about backing up and restoring MQSeries for Tandem NSK.

Chapter 34. Security planning for MQSeries for Tandem NSK

This chapter describes how access to MQSeries for Tandem NSK resources is controlled. It contains these sections:

Full details of MQSeries for Tandem NSK security handling are given in the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Controlling access to resources

Access to MQSeries for Tandem NSK queue manager resources is controlled by the *object authority manager (OAM)*, which is the default authorization-service installable component. You can implement your own security controls in place of, or in addition to, those supplied by the OAM. For more information on installable services and installable components, see "Chapter 8. Introduction to the MQSeries Framework" on page 55.

Users can access queue manager objects (queues, process definitions, and queue managers) only if they have the required authority. The OAM manages a user's authorization to manipulate MQSeries objects, and provides a command interface through which you can grant or revoke access authority to an object for a specific group of users.

Managing access through principals and user groups

For each Tandem NSK user ID that needs to access MQSeries, you must define an MQSeries *principal* using the **altmqusr** control command. A principal database provides mappings between MQSeries principals and Tandem NSK user IDs.

Tandem NSK user IDs belong to *user groups*. The OAM manages access to MQSeries resources at the user-group level, not at the level of individual user IDs or principals. The authorizations that a principal has derive from the union of the authorizations of all the groups to which its associated user ID belongs.

Note: A user ID can be associated with a single group only, unless SAFEGUARD is in use.

Managing authorization at the group level reduces the amount of administration required. Typically, the same level of access is required by more than one principal. For example, you might define a group consisting of end users who want to run a particular application. New users can be given access simply by defining an MQSeries principal for each user and adding their Tandem NSK user IDs to the appropriate group.

You are recommended to keep the number of groups as small as possible. For example, you could allocate application users to one group and application administrators to another.

Resources you can protect with the OAM

Through the OAM you can control:

- Access to MQSeries objects through the MQI.

  When an application program attempts to access an object, the OAM checks that the principal associated with the user ID that is making the request has the authorization (through its user group) for the operation requested. In particular, this means that queues, and the messages on queues, can be protected from unauthorized access.
- Permission to use MQSC commands.

  Only members of user group MQM, or those authorized via the **setmqaut** command, can execute queue manager administration commands.

Different groups of users can be granted different kinds of access authority to the same object.

Permission to use control commands and PCF commands is managed by the Tandem NSK operating system rather than by the OAM.

Using the MQSeries security commands

The OAM provides four commands that you can invoke from TACL to manage the authorizations of users. These are:

altmqusr
: Create, remove, or alter an MQSeries principal.

dspmqusr
: Display principal.

setmqaut
: Set or reset authority.

dspmqaut
: Display authority.

Details of these commands can be found in the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Security exits with MQSeries

The message channels that are used for distributed queuing, and the MQI channels that are used between clients and servers, both have security exit facilities that can invoke programs that you have supplied.

For more information on these security exit programs, see the *MQSeries Intercommunication* manual.

Chapter 35. Administration of MQSeries for Tandem NSK

This chapter is a summary of the administration facilities provided by MQSeries for Tandem NSK. It has the following sections:
- "Managing objects with MQSeries"
- "Remote administration with MQSeries" on page 164

Details of the commands, command interfaces, and utilities that are provided by MQSeries for Tandem NSK are given in the *MQSeries for Tandem NonStop Kernel System Management Guide*. You need to arrange for users who need to be able to use these administration facilities to have the necessary authorizations, using the procedures given in the *System Management Guide*.

Managing objects with MQSeries

It is the administrator's job to monitor MQSeries for Tandem NSK and make any changes that might be necessary. To do this, the administrator needs to know where each MQSeries object resides, what its characteristics are, and who has access to it.

The administrator can manage and monitor the resources using MQSeries commands (MQSC), or, if there are sets of commands that are issued regularly, by writing an application program that places them on the command queue.

MQSeries can use the security features provided by the OAM, or by a security component that you have installed, to ensure that the user is authorized to issue particular commands for particular resources.

Commands on MQSeries

MQSeries for Tandem NSK supports the following administration commands and facilities:

- You can enter control commands on the command line.
- You can use the **runmqsc** control command to cause MQSC commands from standard input to be executed.
- Any local or remote MQSeries application program can generate PCF commands in messages and put them to the command queue SYSTEM.ADMIN.COMMAND.QUEUE, to be processed by the MQSeries for Tandem NSK command server.

In addition:

- Some TS/MP (Pathway) commands are used for administration.
- The MQM (Message Queue Management) facility, which you start by entering `run mqmc` from the queue manager's PATHCOM prompt, supports some administration tasks.

More information on how to use all these facilities is given in the *MQSeries for Tandem NonStop Kernel System Management Guide*.

Changes made to the resource definitions of a queue manager using the commands (directly or indirectly) are preserved across restarts of MQSeries for Tandem NSK.

Managing communications with MQSeries

Part of the administrator's role is to ensure that the required communications links are activated, and to monitor the status of these links as required by your enterprise. You can find information describing these tasks in the *MQSeries Intercommunication* manual.

Remote administration with MQSeries

There are two aspects to the MQSeries remote administration facilities:
- MQSeries for Tandem NSK can be used to manage remote systems.
- Other remote products can be used to manage MQSeries for Tandem NSK.

Managing remote systems from MQSeries

Facilities are provided by MQSeries for Tandem NSK to allow an administrator to manage the following remote systems:
- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/2 Warp
- MQSeries for OS/390
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Because of the differences in the MQSeries products, it is not always possible to manage remotely the same set of MQSeries objects or attributes that you can manage locally.

If you want to manage any other MQSeries product, you can write an application program to send the appropriate commands to the command queue for that product. However, some MQSeries products do not have a command queue, so they cannot accept commands from local or remote application programs.

Managing MQSeries from remote systems

MQSeries for Tandem NSK can be managed from a remote MQSeries system, by an administrator using the facilities provided by the following products:
- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/390
- MQSeries for OS/2 Warp
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Note: You can manage MQSeries for Tandem NSK from MQSeries for OS/390 by writing an application program to send the appropriate PCF commands to the command queue.

Storage and capacity planning information

For latest storage and capacity-planning information, see SupportPac MPT1 (MQSeries for Tandem NonStop Kernel V2.2.0.1 Performance Report) on the MQSeries Web site at:

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

Part 7. Planning for MQSeries on UNIX systems

Chapter 36. Introduction to MQSeries for UNIX systems

MQSeries products that run on UNIX operating systems comprise two parts, the *server* and the *clients*. The server runs on a machine that is capable of running an MQSeries queue manager on a UNIX system; the clients provided with the product are as follows:

- The MQSeries clients for AIX, HP-UX, Sun Solaris, OS/2 Warp, Windows 3.1, DOS, Windows 95, Windows 98, and Windows NT are provided with MQSeries for AIX, MQSeries for HP-UX, and MQSeries for Sun Solaris.
- The MQSeries clients for OS/2 Warp, Windows 3.1, DOS, Windows 95, and the appropriate UNIX platform are provided with MQSeries for AT&T GIS UNIX and MQSeries for SINIX and DC/OSx.
- The MQSeries client for DIGITAL UNIX (Compaq Tru64 UNIX) is provided with MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX).

A variety of terminals (such as X-stations, nonprogrammable terminals, and portables) can access the server code. There is no MQSeries code on such terminals.

For more information on the hardware and software environments, see:
- "MQSeries for AIX" on page 230
- "MQSeries for AT&T GIS UNIX" on page 234
- "MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX)" on page 259
- "MQSeries for HP-UX" on page 237
- "MQSeries for SINIX and DC/OSx" on page 243
- "MQSeries for Sun Solaris" on page 245

In this section, you are sometimes referred to other books for more information; the appropriate books are:

Table 14. MQSeries on UNIX systems: system administration manuals

UNIX platform	**See these manuals**
AIX	*MQSeries for AIX V5.1 Quick Beginnings, MQSeries System Administration*
AT&T GIS UNIX	*MQSeries for AT&T GIS UNIX System Management Guide*
DIGITAL UNIX (Compaq Tru64 UNIX)	*MQSeries for Digital UNIX System Management Guide*
HP-UX	*MQSeries for HP-UX V5.1 Quick Beginnings, MQSeries System Administration*
SINIX and DC/OSx	*MQSeries for SINIX and DC/OSx System Management Guide*
Sun Solaris	*MQSeries for Sun Solaris V5.1 Quick Beginnings, MQSeries System Administration*

Planning for MQSeries

This chapter helps you to plan for the introduction of MQSeries on UNIX systems into your enterprise, and introduces the items that you need to consider when doing this planning.

There are several stages in planning for the use of MQSeries on UNIX systems that you must go through. They are:

1. Preparing your applications for the use of MQSeries on UNIX systems
2. Planning to include MQSeries on UNIX systems in a network
3. Preparing to install MQSeries on UNIX systems
4. Planning to set up MQSeries on UNIX systems

A prime requirement for a message delivery system is that it must be reliable. Many functions are built into MQSeries on UNIX systems to ensure that:

- Messages are not lost despite system failures.
- Messages are not delivered more than once.
- Messages are not accessed by, or delivered to, unauthorized persons or applications.

MQSeries on UNIX systems uses logging and other facilities to support these functions.

You must also plan for the operation and administration of MQSeries on UNIX systems in your enterprise, and consider the implementation of an appropriate set of security facilities. Brief outlines of these planning operations are included in this chapter.

Preparing your applications for use with MQSeries

MQSeries on UNIX systems brings the Message Queue Interface (MQI) to your applications. This interface allows you to modify existing applications and to write new applications. The MQI removes much of the need to understand the network and communication systems that you use. Thus you can expect to generate applications more speedily than before. However, you must prepare to take advantage of the MQI by planning its use in your applications and by understanding the ways in which it assists you.

You can find more information on how to use the MQI in your applications by referring to the *MQSeries Application Programming Guide*.

MQSeries for AT&T GIS UNIX, DIGITAL UNIX (Compaq Tru64 UNIX), and DC/OSx do not support threads. MQSeries for AIX, HP-UX, SINIX, and Sun Solaris support multithreaded applications, but there are limitations that are documented in the *MQSeries Application Programming Guide*. If the platform that you intend to use is not specifically mentioned, check with your IBM support center whether multithreaded applications are supported.

Interfacing with CICS with MQSeries

With MQSeries products on UNIX systems, you can create application programs for the appropriate CICS transaction environment. These applications can use the MQI to communicate with CICS or non-CICS programs in any of the environments supported by the MQSeries products.

Planning to use MQSeries in a network

MQSeries on UNIX systems uses the distributed queue management (DQM) facility to exchange messages between MQSeries platforms, using either the SNA LU 6.2 or TCP transmission protocols. The UDP (User Datagram Protocol) can also be used on AIX.

You must consider how you will attach MQSeries on UNIX systems to a network, and how you will define the message channels that will be used to exchange messages. You can simplify your definitions for distributed queuing on AIX, HP-UX, and Sun Solaris by using MQSeries clusters.

According to the way that you have set your systems up, security checks can be performed at various times. MQSeries products on UNIX systems do not provide communications link authorization or data encryption on these links. Instead, various exits are provided that can be used by your applications to provide these facilities. On AIX, HP-UX, and Sun Solaris, MQSeries also provides exits relating to DCE security.

"MQSeries interoperability summary" on page 223 shows the links that are possible to other MQSeries products, and the transmission protocols that can be used.

For further information about distributed queue management, refer to the *MQSeries Intercommunication* and the *MQSeries Queue Manager Clusters* manuals.

Installing MQSeries

Before you can install one of the MQSeries products on a UNIX system, you must create both a group and user with the name **mqm**, that will own the directories and files that contain the various resources associated with the product. These are created for you when you install MQSeries on AIX, provided you are not using NIS (if they do not already exist).

To prepare for the actual installation, you need to plan how much disk space will be required in your UNIX system to accommodate MQSeries. Assistance is given in "Chapter 40. Storage planning for MQSeries on UNIX systems" on page 183 to help you plan the amount of space required.

You can find more information to help you with the installation of MQSeries on a UNIX system in the appropriate manual (see Table 14 on page 167).

Setting up MQSeries

After installation, MQSeries needs to be set up, and customized for your own use. This ensures that the appropriate UNIX system facilities are made available to MQSeries, and that your MQSeries system is correctly initialized and ready to work with your applications.

MQSeries uses configuration files to hold the product configuration information used for logging, communications protocols and installable components. After installing the product, you can edit these files to tailor the operation of the product to meet the requirements of your installation.

In addition, you need to do the following:

- Define queues.
  - Consider your naming conventions for queues.
- Define trigger processes.
- Define remote links.
  - Define associated transmission queues.
  - Consider your naming conventions for remote queues.
  - Consider your naming conventions for channels.

Note: The characters within the names given to all MQSeries objects are case sensitive. Therefore, be very careful when defining the names of objects, to select the appropriate uppercase or lowercase characters.

You can find more information about the setting up and customizing processes for MQSeries on UNIX systems in the appropriate manual (see Table 14 on page 167).

Planning recovery services with MQSeries

MQSeries provides logging services to allow backup and recovery of the messaging system. “Chapter 37. Backup and recovery planning for MQSeries for UNIX systems” on page 173 introduces you to these facilities, and to the items that you need to consider in order to include MQSeries on UNIX systems in your backup and recovery plans.

You can find more information on the backup and recovery facilities provided by MQSeries in the appropriate manual (see Table 14 on page 167).

Planning data security on MQSeries

MQSeries uses the facilities of the MQSeries *object authority manager (OAM)* installable component to control access to the various different types of queue manager resource (queues, process definitions, channels, and queue managers).

You can find more general information on authorization installable components in “Chapter 8. Introduction to the MQSeries Framework” on page 55.

“Chapter 38. Security planning for MQSeries for UNIX systems” on page 177 introduces you to the security facilities provided by MQSeries and to some of the items you need to consider when planning for security.

You can find more information on the security facilities provided by MQSeries on UNIX systems in the appropriate manual (see Table 14 on page 167).

Administration with MQSeries

A summary of the administration facilities provided is given in “Chapter 39. Administration of MQSeries on UNIX systems” on page 181. Full details of these facilities can be found in the appropriate manual (see Table 14 on page 167).

Support for R/3 with MQSeries

The following products:
- MQSeries link for R/3 for AIX
- MQSeries link for R/3 for DIGITAL UNIX (Compaq Tru64 UNIX)
- MQSeries link for R/3 for HP-UX
- MQSeries link for R/3 for Sun Solaris

provide an interface that enables you to integrate your R/3 application with applications running in other environments (including those on R/3 and R/2 environments).

The R/3 link works with the Application Link Enabling (ALE) layer of the R/3 system to transmit Intermediate Documents (IDocs) into and out of your R/3 system, using MQSeries messages and queues to carry the information. It extends the scope of your business by allowing you to link your R/3 applications to any other application that you can access through MQSeries, even when those applications require different data formats.

For more information, see the *MQSeries link for R/3 Version 1.2 User's Guide*.

Migration from MQSeries Version 2

When you have migrated from MQSeries Version 2 to MQSeries Version 5 you will **not** be able to revert to Version 2. You should back up your system before installing the new version. This will enable you to back off the upgrade if necessary. If you do this however, you will not be able to recover the work performed by MQSeries Version 5.

With MQSeries Version 5, the system default objects are created automatically when you use the crtmqm command to create a queue manager. The sample MQSC definition file, AMQSCOMA.TST, is no longer provided. If you have used AMQSCOMA.TST to customize your settings for MQSeries Version 2, and you want to use the same settings with Version 5, save your version of the file before you install MQSeries Version 5. You can then use this file to create the Version 2 default objects. Alternatively, you can generate a new MQSC definition file if required.

A list of the system default objects for MQSeries Version 5 is provided in the *MQSeries System Administration* manual.

Euro symbol support

Support for the new euro currency symbol has been added to MQSeries on UNIX systems, either by inclusion in latest releases or via service.

If you need to modify your applications to use this symbol, ensure that they use one of the coded character sets that include it. These are described in the *MQSeries Application Programming Reference* manual. If you need to change the coded character set used by your queue manager, use the CCSID attribute of the ALTER QMGR command (or the equivalent PCF command).

Chapter 37. Backup and recovery planning for MQSeries for UNIX systems

This chapter describes the background concepts of recovery and restart. It contains the following sections:

More details of the logging and recovery facilities of MQSeries on UNIX systems are given in the appropriate manual (see Table 14 on page 167).

Logging with MQSeries

The basic premise of a messaging system is that messages entered into the system are assured of delivery to the destination. One of the ways of ensuring that messages are not lost is to maintain a record of the activities of the queue manager that handles the receipt, transmission, and delivery of messages.

MQSeries on UNIX systems does this by recording all the significant changes to the data controlled or managed by the queue manager in a log. This process is called logging. The data changes that are logged include the puts and gets of persistent messages to and from queues, changes to queue attributes, and channel activity.

The purpose of logging is to create and maintain a log that:

- Keeps records of queue manager changes
- Keeps records of queue updates for use by the restart process
- Is a source for restoration of data should there be a hardware or software failure

Each MQSeries log consists of a log control file, together with one or more log files for the storage of data.

The log control file is, as its name implies, used to control and monitor the use of the log files. It contains information relating to the size, location, next available file, and other data related particularly to the log files themselves. All the log files within one log are the same size; there is a default value for this size, but you can override this value when you set up the log.

On some UNIX platforms, MQSeries provides a log dump facility (DMPMQLOG). This enables you to format and display the contents of the log when doing problem determination.

Types of logging with MQSeries

MQSeries on UNIX systems has two approaches to maintaining records of queue manager activities:

- Circular logging
- Linear logging

Each type of logging stores the recorded data in a set of files. The differences between the two types of logging are the contents, and the way that the files are linked together.

With circular logging, the set of log files are effectively linked together so as to form a ring. When data is collected, it is written sequentially into the files, in such a way as to reuse the log files in the ring. You can use circular logging for:

- Crash recovery - that is, after a system failure of some kind has stopped the queue manager unexpectedly
- Restart recovery - after a planned closedown of the system

With linear logging, the log is maintained as a continuous sequence of files. When data is collected, it is written sequentially into the log files; the space in the files is not reused, so that you can always retrieve any record from the time that the queue manager was created.

Because disk space is finite, you might have to plan for some form of archiving. Also, if you are handling a high volume of persistent messages, all your log files will eventually be filled. This will result in operator messages being written to an error log file; some action will need to be taken by the system administrator to make more log space available, or to reuse the existing space. You can use linear logging for:

- Crash recovery
- Restart recovery
- Media recovery - to recreate lost or damaged data after a media failure by replaying the contents of the log

Selecting a logging method with MQSeries

You must base your selection of log type on your requirements for recovery.

Both types of logging can cope with unexpected power outages in the absence of hardware failure. If you accept that only crash or restart recovery is required, circular logging might be adequate. If media recovery is important to you, select linear logging.

With each type of logging, you need to decide on the number of files to use in the log, and their size. The total amount of space needed depends on the amount of data to be recorded, which depends on various parameters, including:

- The size of messages
- The number of puts and gets from queues
- The number of messages being transmitted by the message channel agents

Resource management with MQSeries

MQSeries on UNIX systems supports the coordination of transactions by an external transaction manager that uses the X/Open XA interface. On some UNIX platforms, MQSeries can also act as a syncpoint coordinator, coordinating updates made by external resource managers.

MQSeries as a resource manager

In an XA configuration, the MQSeries queue manager acts as an XA resource manager, managing message queues. The XA transaction manager coordinates the operations of the queue manager, and any other XA-compliant resource managers, to synchronize the commit or backout of transactions. This ensures that updates to MQSeries message queues are coordinated with the updates to all the other types of resource being managed.

With MQSeries on UNIX systems, the XA transaction manager can be as follows:

CICS/TXSeries
AIX, HP-UX, SINIX, and Sun Solaris

Encina
SINIX

Tuxedo
AIX, AT&T GIS UNIX[2], HP-UX, SINIX and DC/OSx, and Sun Solaris

The MQSeries resources are committed or backed out as directed by CICS, ENCINA, or TUXEDO. This support is available only on the MQSeries on UNIX systems server, and is not available to client applications.

MQSeries as a transaction manager

On some UNIX platforms, MQSeries can act as a transaction manager and coordinate updates made by external resource managers within MQSeries units of work. These external resource managers must comply to the X/Open XA interface.

MQSeries for AIX, MQSeries for HP-UX, and MQSeries for Sun Solaris can act as transaction managers for the following XA-compliant database managers:
- DB2
- Oracle
- Sybase (AIX and Sun Solaris only)

2. This platform has become NCR UNIX SVR4 MP-RAS, R3.0

Recovering from problems with MQSeries

MQSeries can recover from communications failures and power loss incidents. In addition, it is sometimes possible to recover from other types of problem with the MQSeries data, such as inadvertent deletion of a file.

In the case of a communications failure, messages remain on the queues until they are removed by a receiving application. If the message is being transmitted, it remains on the transmission queue until it can be successfully transmitted. To recover from a communications failure, it is normally sufficient simply to restart the channels using the link that failed.

On a restart after your system has lost power, the queue manager restores all the persistent messages that were on the queues to the state that existed just before the power failure, so that no persistent messages are lost; nonpersistent messages are discarded.

There are ways in which an MQSeries object can become unusable, for example due to inadvertent damage. In such situations, you will have to take steps to recover either your complete system or some part of it. The action required depends on when the damage is detected, whether the log method selected supports media recovery, and which object or objects are damaged.

High availability with MQSeries

For those situations where high availability is a requirement, you might wish to consider using MQSeries for AIX with the AIX High Availability Cluster Multi-Processing/6000 (HACMP/6000™) product.

HACMP/6000 offers an efficient way to recover from failures at the AIX server that is running your application programs. The product allows for servers to be configured to provide, for example, automatic transfer to a standby processor if the primary one fails.

Also on AIX, you can use *disk mirroring* to improve the availability of your application data on disk.

Chapter 38. Security planning for MQSeries for UNIX systems

This chapter describes the access control security features in MQSeries on UNIX systems. It contains these sections:
- "Controlling access to resources with MQSeries"
- "Security exits with MQSeries" on page 179

Full details of MQSeries on UNIX systems security handling are given in the *MQSeries System Administration* manual.

Controlling access to resources with MQSeries

With MQSeries on UNIX systems, access to queue manager resources is controlled through the *object authority manager (OAM)*, which is the default authorization installable component. Because the OAM is an installable component, you can implement your own security controls in place of, or in addition to, those supplied by the OAM. For more information on installable services and installable components, see "Chapter 8. Introduction to the MQSeries Framework" on page 55.

Users can access queue manager objects (queues, process definitions, namelists, channels, and queue managers) only if they have the required authority. The OAM manages a user's authorization to manipulate MQSeries objects, and provides a command interface through which you can grant or revoke access authority to an object for a specific group of users.

Managing access through user groups with MQSeries

In discussing security in a UNIX environment, we use the term *principal* rather than user ID. The reason for this is that authorities granted to a user ID can also be granted to other entities, for example, an application program that issues MQI calls, or an administration program that issues PCF commands. In these cases, the principal associated with a program is not necessarily the user ID that was used when the program was started.

Managing access permissions to MQSeries resources is based on *user groups*, that is, groups of principals. The OAM does not maintain authorizations at the level of individual principals. The mapping of principals to group names is carried out within the OAM, and operations are carried out at the group level.

The authorizations that a principal has are the union of the authorizations of all the groups of which it is a member, that is, its *group set*. Whenever a principal requests access to a resource, the OAM computes this union, and then checks the authorization against it.

Resources you can protect with MQSeries

Through MQSeries on UNIX systems you can control:

- **Access to queue manager objects through the MQI**

  When an application program attempts to access an object, the OAM checks to see if the principal making the request has the authorization (through its user group) for the operation requested. In this way, the queues, and the messages on queues, can be protected from unauthorized access.

- **Permission to use MQSeries commands**

  Only members of authorized user groups can execute queue manager administration commands.

  When a member of a user group attempts to execute a command, the OAM checks to see if the principal making the request has the authorization (through its user group) to use the command. These access control checks are performed irrespective of whether the commands are issued through the **runmqsc** (run MQSC commands) command, or through an application that uses PCFs.

Different groups of users can be granted different kinds of access authority to the same object. For example, one group might be allowed to read (**MQGET**) messages from a queue, but not to write (**MQPUT**) messages to that queue. Other groups might be given authority to put messages to and get messages from the same queue. Similarly, some groups might have get and put authority but not be allowed to create or delete queues.

Using groups for authorizations with MQSeries

Using groups for authorization, rather than individual principals, reduces the amount of administration required. Normally, a particular kind of access is required by more than one principal. For example, you might define a group consisting of end users who want to run a particular application. New users can be given access simply by adding their user ID to the appropriate group.

Try to keep the number of groups as small as possible. Dividing principals into one group for application users, and one for administrators, is a good place to start.

Using the security commands with MQSeries

The OAM provides two commands that you can use to manage the authorizations of users. These are:

- **setmqaut** - set or reset authority
- **dspmqaut** - view authority

When you start a queue manager, it uses a group ID of **mqm**, and a user ID of **mqm**. These must be defined before you can start a queue manager. After this, any principal belonging to the group **mqm** can issue **setmqaut** commands to change authorizations to resources.

Details of these commands can be found in the *MQSeries System Administration* manual.

Security exits with MQSeries

The message channels that are used for distributed queuing, and the MQI channels that are used between clients and servers, both have security exit facilities that can invoke programs that you have supplied.

For more information on these security exit programs, see the *MQSeries Intercommunication* manual.

Chapter 39. Administration of MQSeries on UNIX systems

This chapter is a summary of the administration facilities provided by MQSeries on UNIX systems. It has the following sections:

- "Managing objects with MQSeries"
- "Remote administration with MQSeries" on page 182

Details of the commands, command interfaces, and utilities that are provided by MQSeries on UNIX systems are given in the appropriate manual (see Table 14 on page 167).

You need to arrange for users who need to be able to use these administration facilities to have the necessary authorizations, using the procedures given in the appropriate manual (see Table 14 on page 167).

Managing objects with MQSeries

It is the administrator's job to monitor MQSeries on UNIX systems and make any changes that might be necessary. To do this, the administrator needs to know where each MQSeries object resides, what its characteristics are, and who has access to it.

The administrator can manage and monitor the resources using MQSeries commands (MQSC), or, if there are sets of commands that are issued regularly, by writing an application program that places them on the command queue.

MQSeries can use the security features provided by the OAM, or by a security component that you have installed, to ensure that the user is authorized to issue particular commands for particular resources.

Commands on MQSeries

MQSeries supports the following administration commands and facilities:

- You can enter control commands on the command line.
- You can use the **runmqsc** control command to cause MQSC commands from standard input to be executed.
- Any local or remote MQSeries application program can generate PCF commands in messages and put them to the command queue SYSTEM.ADMIN.COMMAND.QUEUE, to be processed by the MQSeries command server.

More information on how to use all these facilities is given in the appropriate manual (see Table 14 on page 167).

Changes made to the resource definitions of a queue manager using the commands (directly or indirectly) are preserved across restarts of MQSeries.

Managing communications with MQSeries

Part of the administrator's role is to ensure that the required communications links are activated, and to monitor the status of these links as required by your enterprise. You can find information describing these tasks in the *MQSeries Intercommunication* manual.

Remote administration with MQSeries

There are two aspects to the MQSeries remote administration facilities:
- The local MQSeries system can be used to manage remote systems.
- Other remote products can be used to manage the local MQSeries system.

Managing remote systems from MQSeries

Facilities are provided by MQSeries to allow an administrator to manage the following remote systems:
- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/390
- MQSeries for OS/2 Warp
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Because of the differences in the MQSeries products, it is not always possible to manage remotely the same set of MQSeries objects or attributes that you can manage locally.

If you want to manage any other MQSeries product, you can write an application program to send the appropriate commands to the command queue for that product. However, some MQSeries products do not have a command queue, so they cannot accept commands from local or remote application programs.

Managing MQSeries from remote systems

The local MQSeries system can be managed from a remote MQSeries system, by an administrator using the facilities provided by the following products:
- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/390
- MQSeries for OS/2 Warp
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Note: You can manage MQSeries on UNIX systems from MQSeries for OS/390 by writing an application program to send the appropriate PCF commands to the command queue.

Chapter 40. Storage planning for MQSeries on UNIX systems

This chapter tells you how to plan the type and amount of storage you require when you include MQSeries on UNIX systems in your network. It has the following sections:

- "RAM considerations for MQSeries"
- "Disk space considerations for MQSeries"
- "Capacity planning and performance figures for MQSeries" on page 184

RAM considerations for MQSeries

The processor memory (RAM) is used by MQSeries on UNIX systems in the execution of the product modules, and as a paging area for the messages that are being processed. Parts of the paging area are written to, and read from, disk as necessary.

The minimum amount of RAM required to run the MQSeries on UNIX systems server is 24 MB; if more RAM is available, the performance of the message processing improves.

The amount of RAM required on each client system for an MQSeries client is small compared to that required for the operating system on each of the platforms.

Disk space considerations for MQSeries

Disk space is used by MQSeries on UNIX systems for the following:

- Product modules - client and server executable modules, the toolkit, and the online documentation
- Paging space - server only
- Message queues - server only
- Logs - server only

Product modules for MQSeries

The disk space that you require for the product modules depends on the options that you decide to install: the options are described in the appropriate manual (see Table 14 on page 167).

The product modules can be stored on a LAN server, as an alternative to them being stored on a disk attached to the client or server. MQSeries loads the modules from the LAN server when required.

Message queues for MQSeries

In order to estimate the total amount of storage that you will need for queues, you need to know:

- The number of queues that you have.
- The maximum number of messages there will be on each of the queues at any one time.

- The average size of message on each of the queues. The amount of storage required for one message varies. It is based on the size of the message data plus the size of the message header (456 bytes), rounded up to the nearest 512-byte block. If you are using distribution lists, or grouped or segmented messages, the size of the header will increase for the transmission queue.

Given these values, you can calculate the total amount of space required for queues. However, this value is likely to be an approximate value only, and it is advisable to add a contingency value, to avoid the situation where there is no space left for messages on the queues when your application is running.

The maximum size of a single queue on MQSeries for AIX, HP-UX, and Sun Solaris is 2 GB; on other UNIX platforms it is 320 MB.

Log files for MQSeries

Significant events and data changes can be logged in circular or sequential logs. In particular, the logs are used for recording persistent messages.

All the log files in a log are of the same size. By default, this size is 4 MB, but this value can be changed by the system administrator when the log is defined.

For a circular log, the system administrator needs to specify how many files should be included in the log. For a sequential log, the number of files will increase over time, until the system administrator archives some of the files, and disposes of them. You need to plan for the permanent storage (diskettes, tape, or other media supported in your enterprise) that is to be used for these archived files.

Capacity planning and performance figures for MQSeries

Information about MQSeries performance and capacity planning is available on the Internet at:

`http://www.ibm.com/software/ts/mqseries/txppacs/txpm1.html`

For AT&T GIS UNIX, DIGITAL UNIX (Compaq Tru64 UNIX), and SINIX and DC/OSx, use the information about MQSeries for AIX as a starting point.

Part 8. Planning for MQSeries for VSE/ESA

Chapter 41. Introduction to MQSeries for VSE/ESA

MQSeries for VSE/ESA runs on any machine that is capable of running the VSE/ESA operating system. It runs as a CICS task, so various features of the product are controlled by CICS itself.

MQSeries provides the MQI programming interface for use by application programs that are running on VSE/ESA.

For information on the hardware and software environments, see "MQSeries for VSE/ESA" on page 249.

Planning for MQSeries

This chapter helps you to plan for the introduction of MQSeries for VSE/ESA into your enterprise, and introduces the items that you need to consider when doing this planning.

There are several stages in planning for the use of MQSeries for VSE/ESA that you must go through. They are:

1. Preparing your applications for the use of MQSeries for VSE/ESA
2. Planning to include MQSeries for VSE/ESA in a network
3. Preparing to install MQSeries for VSE/ESA
4. Planning to set up MQSeries for VSE/ESA

You must also plan for the operation and administration of MQSeries for VSE/ESA in your enterprise, and consider the implementation of an appropriate set of security facilities. Brief outlines of these operations are included in this chapter.

Preparing your applications for use with MQSeries

MQSeries for VSE/ESA brings the Message Queue Interface (MQI) to your applications. This interface allows you to modify existing applications and to write new applications. The MQI removes much of the need to understand the network and communication systems that you use. Thus you can expect to generate applications more speedily than before. However, you must prepare to take advantage of the MQI by planning its use in your applications and by understanding the ways in which it assists you.

You can find more information on how to use the MQI in your applications by referring to the *MQSeries Application Programming Guide*.

Planning to use MQSeries in a network

MQSeries for VSE/ESA uses the distributed queue management (DQM) facility to exchange messages between MQSeries platforms, using either the SNA LU 6.2 or TCP transmission protocols.

You must consider how you will attach MQSeries for VSE/ESA to a network, and how you will define the message channels that will be used to exchange messages. "MQSeries interoperability summary" on page 223 shows the links that are possible to other MQSeries products, and the transmission protocols that can be used.

For further information about distributed queue management, refer to the *MQSeries Intercommunication* manual.

Installing MQSeries

MQSeries for VSE/ESA is installed using standard VSE/ESA installation techniques. For information about installing MQSeries, see the *MQSeries for VSE/ESA System Management Guide*.

To prepare for installation, you need to plan how much disk space will be required in your VSE/ESA system to accommodate MQSeries. The disk space requirements for MQSeries for VSE/ESA are:

3380 3 cylinders

3390 2 cylinders

FBA 4500 blocks

Setting up MQSeries

After installation, MQSeries for VSE/ESA needs to be set up, and customized for your own use. This ensures that the appropriate VSE/ESA facilities are made available to MQSeries, and that your MQSeries system is correctly initialized and ready to work with your applications.

You need to do the following:

- Allocate and initialize the required MQSeries files.
  - Create the setup file.
  - Create the MQSeries configuration file.
  - Create cluster definitions for MQSeries queues.
- Prepare CICS for MQSeries.
  - Define CICS resources into the CICS CSD.
  - Define entry definitions for the CICS Destination Control Table.
  - Define entry definitions for the CICS File Control Table.
- Define remote links.
  - Define associated transmission queues.
  - Consider your naming conventions for remote queues.
  - Consider your naming conventions for channels.

You can find more information on the setting up and customizing processes for MQSeries for VSE/ESA in the *MQSeries for VSE/ESA System Management Guide*.

Backup and recovery planning for MQSeries for VSE/ESA

MQSeries uses its own recovery and restart logic in conjunction with CICS file management. You must define all the MQSeries VSAM clusters in the DFHFCT with the LOG parameter set to YES. In addition, you must ensure that the CICS logging facility is activated with `JCT=xx` or `YES` in your DFHSIT.

If you do not do this, you might lose messages, or MQSeries might not be able to use message sequence numbers correctly.

You can find more information on the recovery facilities provided by MQSeries for VSE/ESA in the *MQSeries for VSE/ESA System Management Guide*.

Security planning for MQSeries for VSE/ESA

MQSeries for VSE/ESA uses CICS to provide its security features.

Administration of MQSeries for VSE/ESA

There are two ways of administering an MQSeries for VSE/ESA system:

- You can use the CICS transaction MQMT.

  MQMT allows you to configure, operate, and monitor an MQSeries for VSE/ESA system. MQMT also supports the browsing of message queues.
- You can use the MQSeries Command Line interface (MQCL).

  MQCL supports the management of queues and channels.

Full details of these facilities can be found in the *MQSeries for VSE/ESA System Management Guide*.

Migration from MQSeries Version 1.4

To use MQSeries application programs that were written for MQSeries Version 1.4 with MQSeries for VSE/ESA, you need to do the following:

1. Redefine all message queues.
2. Redefine all message channels.
3. Recompile the application programs, using the MQSeries Version 2 header files.

This might be a suitable time to consider whether you need to re-design any parts of your application, to take advantage of the additional function provided by MQSeries for VSE/ESA.

One difference between MQSeries for VSE/ESA Version 2.1 or later, and the earlier products is that MQSeries for VSE/ESA does its own queue storage management. It is not necessary to run a utility program to recover the space that was occupied by messages that have been removed from queues by MQGET calls; this is done automatically by MQSeries.

Part 9. Planning for MQSeries on Windows NT

Chapter 42. Introduction to MQSeries for Windows NT

Several MQSeries products run on Microsoft® Windows NT. These products are:

- MQSeries for Windows NT Version 5.1, which runs on Windows NT Version 4.0.
- MQSeries for Windows Version 2.1, which runs on Windows NT Version 4.0 (and Windows 95 and Windows 98). This 'lightweight' messaging and queuing product is described in "Chapter 47. Introduction to MQSeries for Windows" on page 213.
- MQSeries link for R/3 Version 1.2 for Windows NT, which runs on Windows NT Version 3.5.1 and Version 4.0; see "Support for R/3 with MQSeries" on page 196.

In addition, MQSeries for Windows NT V5.1, with CSD03 applied, runs on Microsoft Windows 2000.

The following chapters provide information about MQSeries for Windows NT Version 5.1.

The MQSeries for Windows NT products are in two parts, the *server* and the *clients*. The server runs on a machine that is capable of running Windows NT; the clients provided with the product are for AIX, DOS, HP-UX, OS/2 Warp, Sun Solaris, Windows NT, Windows 3.1, Windows 95, and Windows 98.

MQSeries provides the MQI programming interface for use by application programs that are running on the server or the client processor. More detail on clients and servers is given in "Chapter 7. Introduction to MQSeries clients and servers" on page 49.

MQSeries for Windows NT Version 5.1 automatically installs the system upgrades so that you can use the Microsoft Management Console interface and Active Directory Service Interfaces provided in MQSeries for Windows NT Version 5.1. If you prefer, you can still use the MQSeries command-line interface to Windows NT.

The remaining sections of this chapter give basic information about:

- "Planning for MQSeries"
- "Support for R/3 with MQSeries" on page 196
- "Migration from previous versions of MQSeries" on page 196

For more information on the hardware and software environments needed by MQSeries, see "MQSeries for Windows NT" on page 253.

Planning for MQSeries

This chapter helps you to plan for the introduction of MQSeries into your enterprise, and introduces the items that you need to consider when doing this planning.

There are several stages in planning for the use of MQSeries that you must go through. They are:

1. Preparing your applications for the use of MQSeries
2. Planning to include MQSeries in a network
3. Preparing to install MQSeries
4. Planning to set up MQSeries

A prime requirement for a message delivery system is that it must be reliable. Many functions are built into MQSeries to ensure that messages are not lost despite system failures, and are not delivered more than once. MQSeries uses logging and other facilities to support these functions.

You must also plan for the operation and administration of MQSeries in your enterprise and consider the implementation of an appropriate set of security facilities. Brief outlines of these planning operations are included in this chapter.

Preparing your applications for the use of MQSeries

MQSeries brings the Message Queue Interface (MQI) to your applications. This interface allows you to modify existing applications and to write new applications. The MQI removes much of the need to understand the network and communication systems that you use. Thus you can expect to generate applications more speedily than before. However, you must plan to take advantage of the MQI by planning its use in your applications and by understanding the ways in which it assists you. You can find information on how to use the MQI in your applications in the *MQSeries Application Programming Guide*.

MQSeries applications can exploit Lightweight Directory Access Protocol (LDAP) directories. This is described in the *MQSeries Application Programming Guide*.

Interfacing with CICS with MQSeries

With MQSeries you can create application programs for the CICS for Windows NT transaction environment. These applications can use the MQI to communicate with CICS or non-CICS programs in any of the environments supported by the MQSeries products.

Planning to use MQSeries in a network

MQSeries uses the distributed queuing facility to exchange messages between MQSeries platforms, using the TCP, SNA LU 6.2, SPX, or NetBIOS transmission protocols.

You must consider how you will attach MQSeries to a network, and how you will define the channels that will be used to exchange messages. You can simplify your definitions for distributed queuing by using MQSeries clusters.

According to the way that you have set your systems up, security checks can be performed at various times. You can provide your own security services, using the exit facilities provided by MQSeries. MQSeries also provides exits relating to DCE security.

“MQSeries interoperability summary” on page 223 shows the links that are possible to other MQSeries products, and the transmission protocols that can be used.

For further information about distributed queuing, refer to the *MQSeries Intercommunication* and the *MQSeries Queue Manager Clusters* manuals.

Installing MQSeries

Information on installing MQSeries for Windows NT is given in the *MQSeries for Windows NT V5.1 Quick Beginnings* manual.

When you have installed MQSeries, you can run the supplied PostCard application to check that the product has installed correctly.

Setting up MQSeries

After installation, MQSeries needs to be set up, and customized for your own use. This ensures that the appropriate operating system facilities are made available to MQSeries, and that your MQSeries system is correctly initialized and ready to work with your applications.

MQSeries for Windows NT uses the registry to hold the product configuration information used for logging, communications protocols, and installable components.

You then use the MQSeries Explorer user interface to define the queues, queue managers, and other objects that will be needed in your MQSeries network.

Notes:

1. The characters within the names given to all MQSeries objects are case sensitive. Be very careful when defining the names of objects to select the appropriate uppercase or lowercase characters.
2. You should consider your naming conventions for:
   - Queues
   - Remote queues
   - Channels

You can find more information on the setting up and customizing processes for MQSeries in “Chapter 45. Administration of MQSeries for Windows NT” on page 205.

Planning recovery services for MQSeries

MQSeries provides logging services to allow backup and recovery of the messaging system. “Chapter 43. Backup and recovery planning for MQSeries for Windows NT” on page 199 introduces you to these facilities, and to the items that you need to consider in order to include MQSeries in your backup and recovery plans.

You can find more information on the backup and recovery facilities provided by MQSeries in the *MQSeries System Administration* manual.

Planning data security for MQSeries

MQSeries provides a number of security facilities for use by your applications. An introduction to these facilities is given in “Chapter 44. Security planning for MQSeries for Windows NT” on page 203.

You can find more information on the security facilities provided by MQSeries in the *MQSeries System Administration* manual.

Administration of MQSeries

A summary of the administration facilities provided by MQSeries is given in "Chapter 45. Administration of MQSeries for Windows NT" on page 205.

Support for R/3 with MQSeries

The MQSeries link for R/3 for Windows NT product provides an interface that enables you to integrate your R/3 application with applications running in other environments (including those on R/3 and R/2 environments).

The R/3 link works with the Application Link Enabling (ALE) layer of the R/3 system to transmit Intermediate Documents (IDocs) into and out of your R/3 system, using MQSeries messages and queues to carry the information. It extends the scope of your business by allowing you to link your R/3 applications to any other application that you can access through MQSeries, even when those applications require different data formats.

For more information, see the *MQSeries link for R/3 Version 1.2 User's Guide*.

Migration from previous versions of MQSeries

When you have migrated from MQSeries Version 2 to MQSeries Version 5 you will **not** be able to revert to Version 2. Similarly, if you have migrated from Version 5.0 to Version 5.1, you will not be able to revert to Version 5.0. You should back up your system before installing the new version. This will enable you to back off the upgrade if necessary. If you do this however, you will not be able to recover the work performed by the later version of MQSeries.

With MQSeries Version 5, the system default objects are created automatically when you use the **crtmqm** command to create a queue manager.

The sample MQSC definition file, AMQSCOMA.TST, is no longer provided. If you have used AMQSCOMA.TST to customize your settings for MQSeries Version 2, and you want to use the same settings with Version 5, save your version of the file before you install MQSeries Version 5. You can then use this file to create the Version 2 default objects. Alternatively, you can generate a new MQSC definition file if required.

A list of the system default objects for MQSeries Version 5 is provided in the *MQSeries System Administration* manual.

With MQSeries Version 5.1, the installation process decides whether this is a new installation or an update from a previous level of MQSeries for Windows NT. If you are moving from a earlier level, all the files you previously selected (for example, the toolkit) or created (for example, your queue managers) are maintained.

Note: MQSeries for Windows NT Version 5.1 automatically migrates configuration information from your .INI files into the Windows NT registry. Configuration information is then updated when you define or change details through the user interface.

Euro symbol support

Support for the new euro currency symbol has been added to MQSeries. If you need to modify your applications to use this symbol, ensure that they use one of the coded character sets that include it. These are described in the *MQSeries Application Programming Reference* manual. If you need to change the coded character set used by your queue manager, use the CCSID attribute of the ALTER QMGR command (or the equivalent PCF command).

Chapter 43. Backup and recovery planning for MQSeries for Windows NT

This chapter describes the background concepts of recovery and restart. It contains the following sections:
- "Logging with MQSeries"
- "Resource management with MQSeries" on page 200
- "Recovering from problems with MQSeries" on page 201

More details of the logging and recovery facilities are given in the *MQSeries System Administration* manual.

Logging with MQSeries

The basic premise of a messaging system is that messages entered into the system are assured of delivery to the destination. One of the ways of ensuring that messages are not lost, is to maintain a record of the activities of the queue manager that handles the receipt, transmission, and delivery of messages.

MQSeries does this by recording all the significant changes to the data controlled or managed by the queue manager in a log. This process is called *logging*. The data changes that are logged include the puts and gets of persistent messages to and from queues, changes to queue attributes, and channel activities.

The purpose of logging is to create and maintain a log that:
- Keeps records of queue manager changes
- Keeps records of queue updates for use by the restart process
- Is a source for restoration of data should there be a hardware or software failure

Each MQSeries log consists of a log control file, together with one or more log files for the storage of data.

The log control file is, as its name implies, used to control and monitor the use of the log files. It contains information relating to the size, location, next available file, and other data related particularly to the log files themselves. All the log files within one log are the same size; there is a default value for this size, but you can override this value when you set up the log.

MQSeries provides a log dump facility (DMPMQLOG). This enables you to format and display the contents of the log when doing problem determination.

Types of logging with MQSeries

In MQSeries, there are two approaches to maintaining records of queue manager activities:
- Circular logging
- Linear logging

Each type of logging stores the recorded data in a set of files. The differences between the two types of logging are the contents, and the way that the files are linked together.

With circular logging, the set of log files are effectively linked together so as to form a ring. When data is collected, it is written sequentially into the files in such a way as to reuse the log files in the ring. You can use circular logging for:

- Crash recovery - that is, after a system failure of some kind has stopped the queue manager unexpectedly
- Restart recovery - after a planned close down of the system

With linear logging, the log is maintained as a continuous sequence of files. When data is collected, it is written sequentially into the log files; the space in the files is not reused, so that you can always retrieve any record from the time that the queue manager was created.

Because disk space is finite, you might have to plan for some form of archiving. Also, if you are handling a high volume of persistent messages, all your log files will eventually be filled. This will result in operator messages being written to an error log file, and some action will need to be taken by the system administrator to make more log space available, or to reuse the existing space. You can use linear logging for:

- Crash recovery
- Restart recovery
- Media recovery - to recreate lost or damaged data after a media failure by replaying the contents of the log

Selecting a logging method with MQSeries

You must base your selection of log type on your requirements for recovery.

Both types of logging can cope with unexpected power outages in the absence of hardware failure. If you accept that only crash or restart recovery is required, circular logging might be adequate. If media recovery is important to you, select linear logging.

With each type of logging, you will need to decide on the number of files to use in the log, and their size. The total amount of space needed will depend on the amount of data to be recorded, which depends on various parameters, including:

- The size of messages
- The number of gets and puts to queues
- The number of messages being transmitted by the message channel agents

Resource management with MQSeries

MQSeries supports the coordination of transactions by an external transaction manager that uses the X/Open XA interface. MQSeries can also act as a transaction manager, coordinating updates made by external resource managers.

MQSeries as a resource manager

MQSeries supports the coordination of transactions by an external transaction manager that uses the X/Open XA interface.

In an XA configuration, the MQSeries queue manager acts as an XA resource manager, managing message queues. The XA transaction manager coordinates the operations of the queue manager, and any other resource managers, to synchronize the commit or backout of transactions. This ensures that updates to MQSeries message queues are coordinated with the updates to all the other types of resource being managed.

TXSeries for Windows NT operates as an XA transaction manager, so the XA resource management capabilities of MQSeries for Windows NT can be used. MQSeries for Windows NT can also use BEA Tuxedo as a transaction manager.

This support is available only on the MQSeries server, and is not available to client applications. See the information on transactional support in the *MQSeries System Administration* manual for more details.

MQSeries as a transaction manager

MQSeries can act as a transaction manager and coordinate updates made by external resource managers within MQSeries units of work. These external resource managers must comply to the X/Open XA interface. MQSeries can act as a transaction manager for DB2 and Sybase.

Recovering from problems with MQSeries

MQSeries will recover from both communications failures and power loss incidents. In addition, it is sometimes possible to recover from other types of problem with the MQSeries data, such as inadvertent deletion of a file.

In the case of a communications failure, messages simply remain on queues until they are removed from the queues by a receiving application. If the message is being transmitted, it remains on the transmission queue until it can be successfully transmitted. To recover from a communications failure, it will normally be sufficient simply to restart the channels using the link that failed.

On a restart after your system has lost power, the queue manager will restore all the persistent messages on the queues to the state that existed just before the power failure, so that no persistent messages are lost; nonpersistent messages are discarded.

There are ways in which an MQSeries object can become unusable, for example due to inadvertent damage. In such situations, you will have to take steps to recover either your complete system or some part of it. The action required depends on when the damage is detected, whether the log method selected supports media recovery, and which object or objects are damaged.

You can configure your queue manager to restart without user intervention when the system powers up. This can include the channel initiator, any listeners, the command server, and the trigger monitor.

Chapter 44. Security planning for MQSeries for Windows NT

In general, each MQSeries product provides security facilities by building on those provided by the platform for which the product was designed.

This chapter describes the following security facilities provided by MQSeries for Windows NT:

User ID support with MQSeries

MQSeries for Windows NT handles the user ID of the logged-on user in two ways:

1. The user ID is stored in the message descriptor of messages when they are created.
2. The user ID is used when MQSeries performs authorization checks for access to MQSeries objects, such as queues.

The maximum length of user ID that is used in these ways by all MQSeries products is 12 characters.

On Windows NT, the user name can be longer than 12 characters. For example, the default user name for administration on Windows NT is *Administrator*.

MQSeries for Windows NT Version 5.1 supports user names that are longer than 12 characters, while at the same time maintaining compatibility with other products in the MQSeries family. The Windows NT security identifier (SID) is used to supplement the 12-character user ID. The SID contains information that identifies the full user account details on the Windows NT security account manager (SAM) database where the user is defined.

When a message is created on MQSeries for Windows NT, MQSeries stores the SID in the message descriptor. When MQSeries for Windows NT performs authorization checks, it uses the SID to query the full information from the SAM database.

Note: The SAM database in which the user is defined must be accessible for this query to succeed.

The Windows NT SID allows MQSeries to distinguish between two user names that are the same. For example, the user name *JohnSmith* can be defined on different Windows NT SAMs. A new option on the Windows NT authorization service controls whether the authority checks require both a user ID and a valid Windows NT SID to be available before granting access to MQSeries objects. For example, when this option is set, administration messages to the SYSTEM.ADMIN.COMMAND.QUEUE from other systems must supply a SID that is valid on the local system and that resolves to the user ID in the message context.

Security planning

The changes in MQSeries for Windows NT Version 5.1 that allow the Windows NT SID to be used are as follows:

- The message queue interface (MQI):
  - Changes to the *AccountingToken* field in the Message Descriptor (MQMD)
  - A new version, MQOD_VERSION_3, of the Object Descriptor (MQOD) to provide the *AltSecurityIdentifier* field

  See the *MQSeries Application Programming Guide* and the *MQSeries Application Programming Reference* manuals for more information.
- New fields in the Channel Definition (MQCD) structure to support longer user ID fields.

  See the *MQSeries Intercommunication* manual for more information.
- The maximum length of the MCAUSER parameter in the DEFINE CHANNEL, ALTER CHANNEL, and DISPLAY CHANNEL MQSC commands is increased to 64 characters.

  See the *MQSeries Command Reference* manual for more information.
- The maximum length of the *MCAUserIdentifier* parameter in the Create Channel, Change Channel, Copy Channel, and Inquire Channel PCF commands is increased to 64 characters.

  See the *MQSeries Programmable System Management* manual for more information.
- The MQSeries Object Authority Manager (OAM), through its **setmqaut** and **dspmqaut** commands, allows a long user name and a domain qualifier.

  See the *MQSeries System Administration* manual for more information.

Security exits with MQSeries

The message channels that are used for distributed queuing, and the MQI channels that are used between clients and servers, both have security exit facilities that can invoke programs you have supplied.

For more information on these security exit programs, see the *MQSeries Intercommunication* manual.

MQAdmin user ID

When MQSeries for Windows NT is installed, a user administrator account called MQAdmin is created for MQSeries on the local machine to run the MQSeries service. The password for this account is generated by MQSeries. If you want to change this password, you must change it in the following two places:

- The **User Manager** for the local machine
- The **Run as** property of the **IBM MQSeries Services** application in DCOMCNFG

This is described in the *MQSeries System Administration* manual.

Chapter 45. Administration of MQSeries for Windows NT

This chapter is a summary of the administration facilities provided by MQSeries. It has the following sections:

Managing objects with MQSeries

It is the administrator's job to monitor MQSeries and make any changes that might be necessary. To do this, the administrator needs to know where each MQSeries object resides, what its characteristics are, and who has access to it.

The administrator can manage and monitor the resources by using:

- The MQSeries Explorer
- The MQSeries Services snap-in
- The MQSeries Web Administration application
- The Active Directory Service Interfaces
- The MQSeries control commands, MQSC commands, or (if there are sets of commands that are issued regularly) by writing an application program that places them on the command queue

Microsoft Management Console snap-in applications

MQSeries for Windows NT provides two snap-in applications that run under the Microsoft management console (MMC). They provide a graphical user interface for administering the elements of your MQSeries network.

MQSeries Explorer

The MQSeries Explorer snap-in application allows you to define and control:

- Queue managers
- Clusters (networks of queue managers that can be on several MQSeries systems)
- Other objects (such as queues, channels, processes, client connections, and namelists)

The user interface provides extensive help information to guide you through the tasks involved.

MQSeries services snap-in

The MQSeries services snap-in application allows you to define and control services (for example, the MQSeries command server, channel initiators, listeners, and configuration information).

The user interface provides extensive help information to guide you through the tasks involved.

Web administration

MQSeries for Windows NT also provides a web-based application that allows you to administer your MQSeries network from a web browser running on Windows NT, Windows 95, or Windows 98. The application allows you to use MQSeries command (MQSC) facilities, either as individual commands or multiple commands in a script.

Its user interface provides extensive help information to guide you through the tasks involved.

Active Directory Service Interfaces

Support for the Microsoft Active Directory Service Interfaces (ADSI) is available through the `IBMMQSeries` namespace. ADSI support within MQSeries supports the modification of existing objects only, you cannot create new ones.

For more information about ADSI, see the *MQSeries for Windows NT Using the Component Object Model Interface* manual.

Commands with MQSeries

MQSeries supports the following administration commands and facilities:

- MQSeries provides control commands that you can enter through the Windows NT command line.
- You can use the **runmqsc** control command to cause MQSC commands from standard input to be executed.
- Any local or remote MQSeries application program can generate PCF commands in messages, and put them to the command queue, SYSTEM.ADMIN.COMMAND.QUEUE, to be processed by the MQSeries command server.

More information on how to use all these facilities is given in the *MQSeries System Administration* manual.

Changes made to the resource definitions of a queue manager using the commands (directly or indirectly) are preserved across restarts of the MQSeries system.

Managing communications with MQSeries

Part of the administrator's role is to ensure that the required communications links are activated, and to monitor the status of these links as required by your enterprise. You can find information describing these tasks in the *MQSeries Intercommunication* manual.

Remote administration with MQSeries

There are two aspects to the MQSeries remote administration facilities:

- MQSeries for Windows NT can be used to manage remote systems.
- Other remote products can be used to manage a MQSeries for Windows NT system.

Managing remote systems from MQSeries

Facilities are provided by MQSeries to allow an administrator to manage the following remote systems:

- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/390
- MQSeries for OS/2 Warp
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Because of the differences between the MQSeries products, it is not always possible to manage remotely the same set of MQSeries objects or attributes that you can manage locally.

If you want to manage any other MQSeries product, you can write an application program to send the appropriate commands to the command queue for that product. However, some MQSeries products do not have a command queue, so they cannot accept commands from local or remote application programs.

Managing MQSeries from remote systems

MQSeries for Windows NT can be managed from a remote MQSeries system, by an administrator using the facilities provided by the following products:

- MQSeries for AS/400
- MQSeries for Compaq (DIGITAL) OpenVMS
- MQSeries for OS/390
- MQSeries for OS/2 Warp
- MQSeries for Tandem NSK
- MQSeries on UNIX systems
- MQSeries for Windows NT

Note: You can manage MQSeries for Windows NT from MQSeries for OS/390 by writing an application program to send the appropriate PCF commands to the command queue.

Chapter 46. Storage planning for MQSeries for Windows NT

This chapter tells you how to plan the type and amount of storage you require when you include MQSeries in your system. It has the following sections:

- "RAM considerations for MQSeries"
- "Disk space considerations for MQSeries"
- "Capacity planning and performance figures for MQSeries" on page 210

RAM considerations for MQSeries

The processor memory (RAM) is used by MQSeries in the execution of the product modules, and as a paging area for the messages that are being processed. Parts of the paging area are written to, and read from, disk as necessary.

The minimum amount of RAM required to run the MQSeries server on Windows NT is 24 MB; if more RAM is available, the performance of the message processing improves. Allow 0.5 MB for each MQSeries client connected.

The amount of RAM required on each client system for an MQSeries client is small compared to that required for the operating system on each of the platforms.

Disk space considerations for MQSeries

Disk space is used by MQSeries for the following:

- Product modules - client and server executable modules, the toolkit, and the online documentation
- Paging space - server only
- Message queues - server only
- Logs - server only

Product modules for MQSeries

The disk space that you require for the product modules depends on the options that you decide to install: the options are described in the *MQSeries for Windows NT V5.1 Quick Beginnings* manual.

Space might be required for client and server executable program modules, the toolkit, and the online documentation.

The product modules can be stored on a LAN server, as an alternative to them being stored on a disk attached to the MQSeries client or server. MQSeries loads the modules from the LAN server when required.

Message queues for MQSeries

In order to estimate the total amount of storage that you will need for queues, you need to know:

- The number of queues that you have.
- The maximum number of messages there will be on each of the queues at any one time.

- The average size of message on each of the queues. The amount of storage required for one message varies. It is based on the size of the message data plus the size of the message header (456 bytes), rounded up the nearest 512-byte block. If you are using distribution lists, or grouped or segmented messages, the size of the header increases for the transmission queue.

Given these values, you can calculate the total amount of space required for queues. However, this value is likely to be an approximate value only, and it is advisable to add a contingency value, to avoid the situation where there is no space left for messages on the queues when your application is running.

The maximum size of a single queue on MQSeries for Windows NT is 2 GB.

Log files for MQSeries

Significant events and data changes can be logged in circular or sequential logs. In particular, the logs are used for recording persistent messages.

All the log files in a log are of the same size. By default, this size is 4 MB, but this value can be changed by the system administrator when the log is defined.

For a circular log, the system administrator needs to specify how many files should be included in the log. For a sequential log, the number of files will increase over time, until the system administrator archives some of the files, and disposes of them. You need to plan for the permanent storage (diskettes, tape, or other media supported in your enterprise) that is to be used for these archived files.

Capacity planning and performance figures for MQSeries

Information about MQSeries performance and capacity planning is available on the Internet at:

`http://www.ibm.com/software/ts/mqseries/txppacs/txpm1.html`

Part 10. Planning for MQSeries for Windows

Chapter 47. Introduction to MQSeries for Windows

MQSeries for Windows is a lightweight messaging and queuing product that provides MQSeries functions on workstations that run on the Microsoft Windows platform. It uses significantly fewer resources than other MQSeries products, so it is a good choice to use as a single-user queue manager running on a small or medium-sized personal computer.

- MQSeries for Windows Version 2.0 is a 16-bit product that runs on Microsoft Windows 3.1 and Windows 95.
- MQSeries for Windows Version 2.1 is a 32-bit product that runs on Windows 95 Windows 98, and Windows NT Version 4.0.

For information about these products, see:
- *MQSeries for Windows V2.0 User's Guide*
- *MQSeries for Windows V2.1 User's Guide*

Do not confuse MQSeries for Windows with MQSeries for Windows NT Version 5. This product is described in "Chapter 42. Introduction to MQSeries for Windows NT" on page 193.

MQSeries for Windows is particularly well suited to users of messaging applications who want to use a standard configuration. It uses definition files that automatically create and start the messaging components the users need, and it can automatically start components when the users start their workstations. These features reduce the need for users of applications to be aware of the messaging product and allow them to concentrate on the applications they want to use.

This chapter contains the following sections:

- "Where to use MQSeries for Windows".

  This explains the intended use of MQSeries for Windows.
- "The features of MQSeries for Windows" on page 215.

  This introduces the features provided by MQSeries for Windows.
- "Comparing queue managers, clients, and servers" on page 216.

  This gives a comparison of supported features on MQSeries for Windows.
- "How MQSeries for Windows differs from the other MQSeries products" on page 217.

  This summarizes the differences between MQSeries for Windows and the other workstation products in the MQSeries family.

Where to use MQSeries for Windows

MQSeries for Windows is designed for use as a *leaf node* in a network of queue managers; that is, it is intended for use by a single user on a workstation that is connected to only one other computer in an MQSeries network of computers (see Figure 14 on page 214).

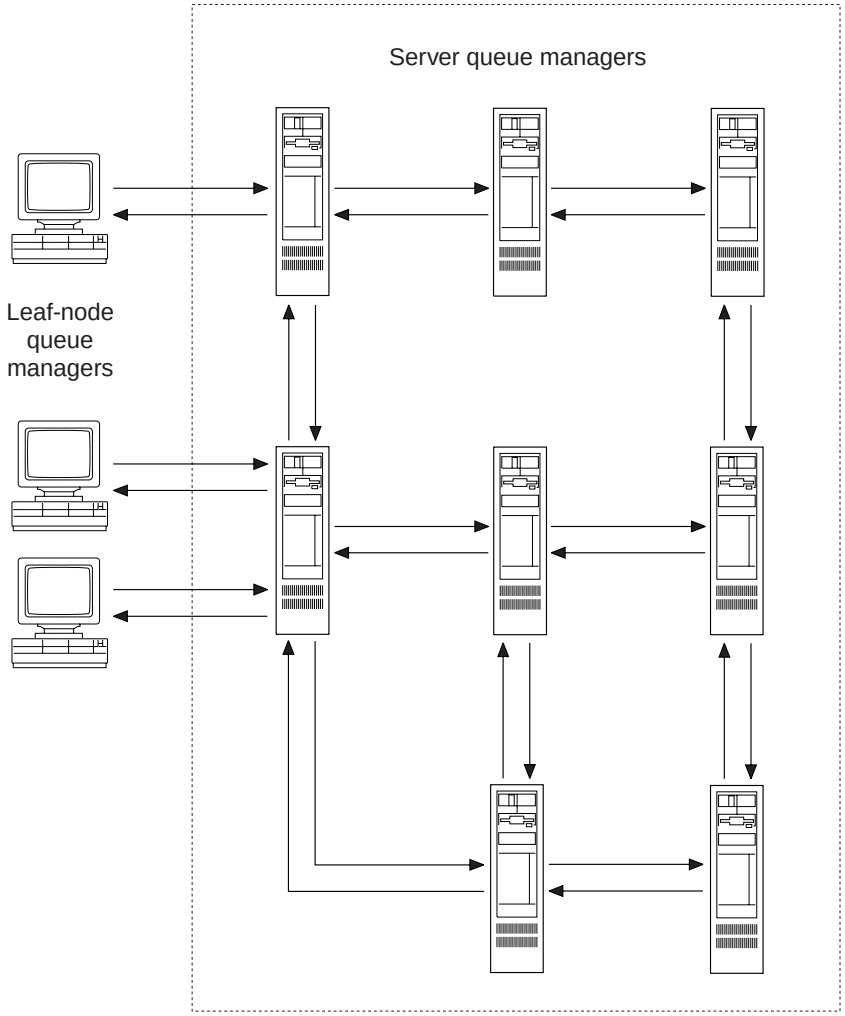

Figure 14. A network of server queue managers and three leaf-node queue managers. The leaf node queue managers run on Windows; they each connect to only one server. The server queue managers run on any other MQSeries platform; they can each have many connections.

There are important differences between a leaf-node queue manager, an MQSeries client, and a server-node queue manager:

- A leaf-node queue manager is a lightweight product that connects to a network of one or more larger servers. It manages its own queues, so an application that runs on a leaf-node queue manager can continue to work, even if there is a failure in the messaging network or if the user decides to work in stand-alone (disconnected) mode (for example, away from their own office or in a branch office that does not currently have a connection to a server).

  A leaf-node queue manager is ***not*** intended for use as an intermediate queue manager that passes messages from one queue manager to another or one that serves many users. For this reason it does not support MQSeries clients. It ***is*** intended for a single user working on their own workstation.
- An MQSeries client provides no queue manager functions and it has no queues. It is dependent on an MQSeries server (of the type that supports MQSeries clients). The server owns the queues that the client uses, so if the communication link between the client and the server is broken, the client cannot use those queues.

- A server-node queue manager is a product (such as MQSeries for Windows NT) that manages the queues and communication channels required to support the transfer of messages between queue managers. The computer on which the server-node queue manager runs is large enough to manage the volume of messages such a server might be required to process, and it might also support MQSeries clients. Such a queue manager is likely to be used by a network administrator.

MQSeries for Windows typically runs on workstations that are not powerful enough to act as a server. Like a server though, MQSeries for Windows manages its own queues and the channels to communicate with other queue managers. However, because it is intended to be a leaf node, MQSeries for Windows does not provide all the server functions available on other MQSeries queue managers; these include media recovery, two-phase commit, instrumentation events, and MQSeries client support. For a full list of the MQSeries features that MQSeries for Windows does not support, see "How MQSeries for Windows differs from the other MQSeries products" on page 217.

MQSeries for Windows is designed to run in the Windows environment, so it provides Windows programs that help to make the queue manager easier to use. These programs are not provided by other MQSeries products.

The features of MQSeries for Windows

MQSeries for Windows provides existing MQSeries features, but on the Windows operating system:

- A small footprint queue manager that runs on Windows
- The MQSeries Message Queue Interface (MQI) for application development on Windows
- Application development support for the C and Visual Basic programming languages
- Communication between queue managers using TCP
- Standard MQSeries message types and formats
- *Persistent messages* (which survive restarts of the workstation) and nonpersistent messages
- Standard MQSC commands to create, alter, or delete MQSeries objects (but MQSeries for Windows does not support all the commands)
- Enablement for automatic installation
- Report generation, including confirm on arrival (COA), confirm on delivery (COD), and message expiry
- Remote administration using PCF commands and MQSeries events (Version 2.1 only)

In addition, MQSeries for Windows provides these features:

- To help users of applications to get started quickly and easily the first time they use the product, MQSeries for Windows uses definition files that automatically create and start the messaging components the users need.
- To help users of applications to get started when they start their workstations, MQSeries for Windows can automatically start its components.

- To help you to set up and work with your MQ™ components:
  - Version 2.0 of MQSeries for Windows provides utilities you can access from the MQSeries for Windows program group in the Windows Program Manager.
  - Version 2.1 of MQSeries for Windows provides an MQSeries Properties dialog box. You can open this from the Windows taskbar and Control Panel.

  These facilities are supported by extensive online help.
- To make it easier to work with the message channels that you must use to send messages between queue managers, MQSeries for Windows provides channel groups. A channel group is a collection of channels that you start and stop at the same time.
- To make it easier to work with dial-up devices (such as modems) when you connect two queue managers:
  - Version 2.0 of MQSeries for Windows provides transport links which can help you to control the duration (and hence the cost) of such a connection.
  - Version 2.1 of MQSeries for Windows uses the dial-up networking connections provided by the operating system.

Comparing queue managers, clients, and servers

If you already use MQSeries clients, see Table 15 for a summary of the differences between an MQSeries for Windows queue manager, an MQSeries client, and an MQSeries server.

Table 15. Comparison of supported features on MQSeries for Windows

Feature	MQSeries for Windows	MQSeries client on Windows	MQSeries for OS/2 Warp
Independent operation	Yes	No	Yes
Queue manager	Yes	No	Yes
Queues	Yes	No	Yes
Message channels	Yes	No	Yes
Run MQSC commands	Utility (V2.0), interactive (V2.1), or command file	No	Command line or command file
Persistence of MQSeries objects	Yes	All objects are on the server	Yes
Logging and media recovery	No	All objects are on the server	Yes
Automatic installation	Yes	Yes	Yes
Automatic start up	Yes	No	No
Supports MQSeries clients	No	Not applicable	Yes

How MQSeries for Windows differs from the other MQSeries products

MQSeries for Windows is a leaf-node queue manager for the Microsoft Windows platform. It is designed to minimize system requirements so that workstations with relatively modest specifications can use commercial messaging. This section summarizes the differences between MQSeries for Windows and the other workstation products in the MQSeries family. The features are listed in alphabetic order.

Attributes of queues and queue managers

MQSeries for Windows does not support all the attributes of queues and queue managers (for example, it does not support those related to triggering). If you use an unsupported attribute in a command or an MQI call, MQSeries for Windows returns a value to show that the attribute is not supported.

Authority checking on the MQOPEN call

MQSeries for Windows does not support the SETMQAUT and DSPMQAUT commands.

Command Server

Version 2.0 of MQSeries for Windows does not support the MQSeries Command Server, so it does not support any MQSeries feature that uses the command server (for example, PCF commands and remote administration).

Version 2.1 provides a command server.

Context passing

MQSeries for Windows does not copy context information from messages it receives from other queue managers. This is because MQSeries for Windows is intended to be a leaf node; it is not intended to be an intermediate node in a network of queue managers, where messages received from one queue manager are passed on to another.

Control commands

In other MQSeries products, you can issue control commands from the command line. MQSeries for Windows provides a user interface to enable you to perform the functions of some of these commands (for example, starting and stopping a queue manager). For a comparison with the MQSeries control commands, see the *MQSeries for Windows V2.1 User's Guide*.

Data conversion

When an MQSeries for Windows queue manager receives data from a queue manager running on a different platform, it cannot convert the machine encoding, integer representation, or coded character set of the application data. Also, it cannot run data conversion exits. This means that any data conversion that is required must be performed by the other queue manager.

Dead-letter queues

MQSeries for Windows does not support dead-letter queues. A dead-letter queue is a queue to which a queue manager or application sends messages it cannot deliver to their correct destination. It is also known as an undelivered-message queue.

An MQSeries for Windows queue manager does not need a dead-letter queue because, being a leaf node, it is always on the edge of a network of

queue managers. This means that it does not have to store messages for onward transmission to other queue managers.

Distributed Computing Environment (DCE) directories
MQSeries for Windows does not support DCE directories.

Distributed Computing Environment (DCE) security
MQSeries for Windows does not support exits relating to DCE security.

Events
See *instrumentation events*.

Installable services
MQSeries for Windows does not support MQSeries installable services. These are additional functions provided in other MQSeries products as several independent components.

Instrumentation events
Instrumentation events are facilities you can use to monitor the operation of queue managers and channels in a network of MQSeries systems. Version 2.1 of MQSeries for Windows generates most of the MQSeries instrumentation events. Version 2.0 does not generate instrumentation events.

Media recovery and logging
MQSeries for Windows does not support the creation of a sequence of log records that contain an image of an object. Other MQSeries products allow you to create such records and re-create objects from this image.

Message Queue Interface (MQI)
MQSeries for Windows supports a subset of the MQI.

To understand those features of the MQI that MQSeries for Windows does not support, see the *MQSeries for Windows V2.1 User's Guide*.

Message retry exit
MQSeries for Windows does not support a message retry exit.

An MQSeries for Windows queue manager does not need a message retry exit because, being a leaf node, it is always on the edge of a network of queue managers. This means that it does not have to store messages for onward transmission to other queue managers.

MQI channels
MQSeries for Windows does not support MQI channels. These are client connection and server connection channels. These are used with MQSeries clients only, so MQSeries for Windows does not support them.

MQSC commands
MQSeries for Windows supports a subset of the MQSC commands. To see which commands it supports, see the *MQSeries for Windows V2.1 User's Guide*.

However, MQSeries for Windows provides facilities that allow you to type MQSC commands in a window (and test and reissue them) and run MQSC command files. This is described in the *MQSeries for Windows V2.1 User's Guide*.

MQSeries client and server support
You cannot use an MQSeries for Windows queue manager as an MQSeries client, nor can you use it to support its own MQSeries clients.

Network support
MQSeries for Windows supports TCP only.

Object Authority Manager (OAM)
MQSeries for Windows does not provide a security manager. It does not support the SETMQAUT and DSPMQAUT commands.

Process definitions
Other MQSeries products use process definitions for setting up the automatic triggering of applications. MQSeries for Windows does not support triggering or process definitions.

Programmable Command Formats (PCFs)
Version 2.0 of MQSeries for Windows does not support PCFs.

Version 2.1 supports many of the PCF commands. To see which commands, see the *MQSeries for Windows V2.1 User's Guide*.

Queue manager
MQSeries for Windows supports multiple queue manager definitions, but it allows only one queue manager to run at any time.

Queue manager quiescing
MQSeries for Windows does not support the quiescing of a queue manager. This is the ability to allow applications to finish processing before the queue manager is stopped, and to prevent any further applications starting.

Sample programs
MQSeries for Windows provides Windows versions of some of the MQSeries sample programs. The MQSeries for Windows samples are described in the *MQSeries for Windows V2.1 User's Guide*.

Security manager
See *object authority manager*.

Signaling
Version 2.0 of MQSeries for Windows does not support signaling.

Version 2.1 supports signalling, so you can use the MQGMO_SET_SIGNAL option with the MQGET call on Windows applications.

Triggering
MQSeries for Windows does not support triggering, so it does not allow a queue manager to start an application automatically when predetermined conditions on a queue are satisfied. The following features of triggering are also not supported:
- Initiation queues
- Process definitions
- Trigger monitors

Two-phase commit
MQSeries for Windows does not support two-phase commit. This is a protocol for the coordination of changes to recoverable resources when more than one resource manager is used by a single transaction.

However, MQSeries for Windows does allow the queue manager to commit or back out units of work.

Part 11. The MQSeries family

Chapter 48. MQSeries product summaries

This chapter gives summary information for all the products in the MQSeries family. It has the following sections:

- "Latest MQSeries products"
- "MQSeries interoperability summary"
- "MQSeries product functional comparison" on page 226

Latest MQSeries products

This book applies to these MQSeries server and client products:

Table 16. MQSeries products

Product name	Server	Client
MQSeries for AIX V5.1	Yes	Yes
MQSeries for AS/400 V5R1	Yes	No
MQSeries for AT&T GIS UNIX V2.2	Yes	Yes
MQSeries for Compaq (DIGITAL) OpenVMS V2.2.1.1	Yes	Yes
MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX) V2.2.1 (PRPQ)	Yes	Yes
MQSeries for HP-UX V5.1	Yes	Yes
MQSeries for OS/2 Warp V5.1	Yes	Yes
MQSeries for OS/390 V2.1 (formerly MQSeries for MVS/ESA)	Yes	No
MQSeries for SINIX and DC/OSx V2.2	Yes	Yes
MQSeries for Solaris on Intel V5.0 (PRPQ)	Yes	Yes
MQSeries for Sun Solaris V5.1	Yes	Yes
MQSeries for Tandem NonStop Kernel V2.2.0.1	Yes	No
MQSeries for VSE/ESA V2.1	Yes	No
MQSeries for Windows V2.0	No	No
MQSeries for Windows V2.1	No	No
MQSeries for Windows NT V5.1	Yes	Yes
MQSeries for Windows NT on DIGITAL Alpha (AXP) V2.0.1 (PRPQ)	Yes	Yes

MQSeries interoperability summary

The tables in this section show the transmission protocols that are supported by the channels in each of the MQSeries products. Tables are provided for the two types of channel that are supported by the MQSeries products:

Message channels
: Used to link MQSeries queue managers.

MQI channels
: Used to link MQSeries clients and servers.

MQSeries interoperability summary

The tables can be used to determine the alternative protocols that can be used to link any two products together, as follows:

1. Find the row in the table that contains the first MQSeries product of interest.
2. Find the row that contains the second MQSeries product of interest.
3. Look at each column position in each of these rows. Where there is a column that has a check symbol ✓ in both rows, then the transmission protocol associated with that column can be used to link the two products.

The "at a glance" sections in "Chapter 49. MQSeries at a glance" on page 229 give details of the prerequisite hardware and software necessary at the two ends of a link for a particular transmission protocol.

Message channels - transmission protocols supported

Table 17. Message channels, transmission protocols supported

MQSeries product	SNA LU 6.2	TCP	NetBIOS	DECnet	SPX
MQSeries for AS/400	✓	✓	—	—	—
MQSeries for AIX	✓	✓	—	—	—
MQSeries for AT&T GIS UNIX	✓	✓	—	—	—
MQSeries for Compaq (DIGITAL) OpenVMS	✓	✓	—	✓	—
MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX)	—	✓	—	—	—
MQSeries for HP-UX	✓	✓	—	—	—
MQSeries for OS/2 Warp	✓	✓	✓	—	✓
MQSeries for OS/390	✓	✓	—	—	—
MQSeries for SINIX and DC/OSx	✓	✓	—	—	—
MQSeries for Solaris on Intel	—	✓	—	—	—
MQSeries for Sun Solaris	✓	✓	—	—	—
MQSeries for Tandem NSK	✓	✓	—	—	—
MQSeries for VSE/ESA	✓	✓	—	—	—
MQSeries for Windows	—	✓	—	—	—
MQSeries for Windows NT	✓	✓	✓	—	✓
MQSeries for Windows NT on DIGITAL Alpha (AXP)	✓	✓	—	—	—

In addition, User Datagram Protocol (UDP) may be used on MQSeries for AIX V5.1 and on MQSeries for Windows V2.0. For both products, UDP support is available in a service release. For further information, contact your IBM representative or see the MQSeries Web site at:

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

MQI channels - transmission protocols supported

Table 18. MQI channels, transmission protocols supported by servers

MQSeries servers	SNA LU 6.2	TCP	NetBIOS	DECnet	SPX
Compaq (DIGITAL) OpenVMS	—	✓	—	✓	—
OS/2 Warp	✓	✓	✓	—	✓
OS/390	✓	✓	—	—	—
OS/400	✓	✓	—	—	—
Solaris on Intel	—	✓	—	—	—
Tandem NSK	✓	✓	—	—	—
UNIX systems	✓(1)	✓	—	—	—
VSE/ESA	✓	✓	—	—	—
Windows NT	✓	✓	✓	—	✓
Windows NT on DIGITAL Alpha (AXP)	✓	✓	—	—	—
Note: (1) Excluding MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX)					

Table 19. MQI channels, transmission protocols supported by clients

MQSeries clients	SNA LU 6.2	TCP	NetBIOS	DECnet	SPX
DOS	—	✓	✓	—	✓
Compaq (DIGITAL) OpenVMS	✓(1)	✓	—	✓	—
OS/2 Warp	✓	✓	✓	—	✓
Solaris on Intel	—	✓	—	—	—
UNIX systems	✓(2)	✓	—	—	—
VM/ESA	✓	✓	—	—	—
Windows NT	✓	✓	✓	—	✓
Windows NT on DIGITAL Alpha (AXP)	✓	✓	—	—	—
Windows 3.1	—	✓	✓	—	✓
Windows 95 and Windows 98	—	✓	✓	—	✓
Notes: (1) On Compaq (DIGITAL) OpenVMS, SNA LU 6.2 supports only PU 2.0. Therefore communication can be to a PU 5.0 host only. (2) Excluding the MQSeries client for DIGITAL UNIX (Compaq Tru64 UNIX).					

MQSeries product functional comparison

Table 20 is a summary of the MQI functions that are provided by the MQSeries products. Those functions that are marked with the symbol ✓ are supported by all products identified by the column heading, except where indicated by the notes following the table.

Table 20. MQSeries product functional comparison

Function	Compaq (DIGITAL) OpenVMS (1)	OS/390	OS/400	Tandem NSK	UNIX systems, OS/2 Warp, Windows NT (1)	VSE/ESA	Windows V2.0 & V2.1
MQCONN/MQDISC	✓	✓	✓	✓	✓	✓	✓
Queue manager groups	✓ (2)	—	—	✓ (2)	✓ (2)	—	—
MQOPEN/MQCLOSE	✓	✓	✓	✓	✓	✓	✓
Queue-manager aliases	✓	✓	✓	✓	✓	✓	✓
Reply-to queue aliases	✓	✓	✓	✓	✓	✓	✓
Default input open option	✓	✓	✓	✓	✓	—	✓
Model/dynamic queues	✓	✓	✓	✓	✓	—	✓
Namelists	—	✓	✓	—	✓ (3)	—	—
Default transmission queue	✓	✓	✓	✓	✓	—	✓
Distribution lists	—	—	✓	—	✓ (3)	—	—
MQPUT	✓	✓	✓	✓	✓	✓	✓
Max message length	4 MB	4 MB	100 MB	4 MB	100 MB(4)	4 MB	4 MB
Max queue size	320 MB	2 GB	2 GB	2 GB(5)	2 GB(6)	4 GB	—
Dead-letter queue	✓	✓	✓	✓	✓	✓	—
Nonpersistent messages	✓	✓	✓	✓	✓	✓	✓
Triggering (first and every)	✓	✓	✓	✓	✓	—	—
Full triggering (depth, priority)	✓	✓	✓	✓	✓	—	—
Message priority	✓	✓	✓	✓	✓	✓	✓
Application-specified syncpoint	✓	✓	✓	✓	✓	✓	✓
Context	✓	✓	✓	✓	✓	✓ (7)	✓ (8)
Exception reports	✓	✓	✓	✓	✓	—	✓
Exception reports with data	✓	✓	✓	✓	✓	—	✓
COA, COD reports	✓	✓	✓	✓	✓	✓	✓
Message expiry	✓	✓	✓	✓	✓	—	✓
Report options for up-level	✓	✓	✓	✓	✓	—	✓
MQGET	✓	✓	✓	✓	✓	✓	✓
Browse	✓	✓	✓	✓	✓	✓	✓
Browse with lock	✓	—	✓	✓	✓	✓	—
Browse under cursor	✓	—	✓	✓	✓	✓	✓
Shared input	✓	✓	✓	✓	✓	✓	✓
Get with signal	—	✓	—	✓	—	—	✓ (10)
Get by MsgId/CorrelId	✓	✓	✓	✓	✓	✓	✓
Message backout count	✓	✓	✓	✓	✓	—	✓
Mark skip backout	—	✓	—	—	—	—	—
Message data conversion	✓	✓	✓	✓	✓	—	—
MQINQ	✓	✓	✓	✓	✓	✓	✓
Queue retention interval	✓	✓	✓	✓	✓	—	—
MQSET	✓	✓	✓	✓	✓	✓	✓
MQCMIT/MQBACK	✓	✓	✓	✓ (16)	✓	—	✓
MQBEGIN	—	—	✓ (15)	—	✓ (3)	—	—
MQSC commands	✓	✓	✓	✓	✓	—	✓

Table 20. MQSeries product functional comparison (continued)

Function	Compaq (DIGITAL) OpenVMS (1)	OS/390	OS/400	Tandem NSK	UNIX systems, OS/2 Warp, Windows NT (1)	VSE/ESA	Windows V2.0 & V2.1
PCF commands	✓	—	✓	✓	✓	—	✓ (10)
API crossing exit	—	✓ (12)	—	—	—	—	—
Message channel exits	✓	✓	✓	✓	✓	—	✓
Events	✓	✓	✓	✓	✓	—	✓ (10)
Segmented messages	—	—	✓	—	✓ (3)	✓	—
Reference messages	—	—	✓	—	✓ (3)	—	—
Fast channels	—	✓	✓	—	✓ (3)	—	✓ (10)
Message retry exit	✓	—	✓	✓	✓	—	—
Channel heartbeats	—	✓ (13)	✓	—	✓ (3)	—	—
Channel auto-definition	—	✓ (14)	✓	—	✓ (3)	—	—
Clusters	—	✓	✓	—	✓ (3)	—	—

Notes:

(1) Functions similar on clients and server.

(2) From client applications only.

(3) AIX, HP-UX, OS/2 Warp, Sun Solaris, and Windows NT only.

(4) On MQSeries for AT&T GIS UNIX, DIGITAL UNIX (Compaq Tru64 UNIX), and SINIX and DC/OSx, the maximum message length is 4 MB.

(5) The maximum queue size is the maximum size allowed by Tandem NSK. The size of a queue file is limited to 2 GB or the available disk space. The queue can be larger than this if the file is partitioned.

(6) On MQSeries for AT&T GIS UNIX, DIGITAL UNIX (Compaq Tru64 UNIX), and SINIX and DC/OSx, the maximum queue size is 320 MB.

(7) Context fields are conveyed without any checks.

(8) Context passing is not supported.

(9) Must do *Browse First and Lock*.

(10) Version 2.1 only.

(11) For queues only.

(12) CICS only.

(13) Not for MQI channels.

(14) Cluster channels only.

(15) On MQSeries for AS/400, queue-manager-coordinated, global units of work are not supported. The OS/400-provided facilities may be used instead.

(16) On Tandem NSK, MQCMIT and MQBACK return MQRC_ENVIRONMENT_ERROR.

MQSeries product functional comparison

Chapter 49. MQSeries at a glance

This chapter gives information about the requirements for each MQSeries platform.

Table 21. MQSeries products at a glance

For information about...	See page...
AIX	230
AS/400	233
AT&T GIS UNIX (1)	234
Compaq (DIGITAL) OpenVMS	235
DOS client	236
HP-UX	237
OS/2 Warp	239
OS/390	241
SINIX and DC/OSx	243
Sun Solaris	245
Tandem NonStop Kernel	247
VM/ESA client	248
VSE/ESA	249
Windows (16-bit)	250
Windows (32-bit)	252
Windows NT	253
Windows 3.1 client	256
Windows 95 and Windows 98 client	257
MQSeries link for R/3	258
MQSeries PRPQ products	259
Note: (1) This platform has become NCR UNIX SVR4 MP-RAS, R3.0	

MQSeries, when used in accordance with its associated documentation, is capable of correctly processing, providing, and receiving date data within and between the twentieth and twenty-first centuries, provided that all products (for example, hardware, software, and firmware) used with this IBM Program properly exchange accurate date data with it.

Customers should contact third party owners or vendors regarding the readiness status of their products.

IBM reserves the right to update the information shown here. For the latest information regarding levels of supported software, refer to

`http://www.ibm.com/software/ts/mqseries/platforms/supported.html`

and for the latest IBM statement regarding Year2000 readiness, refer to

`http://www.ibm.com/IBM/year2000/`

MQSeries for AIX

MQSeries for AIX, Version 5.1

Machine requirements

MQSeries for AIX runs on any IBM RS/6000® that is capable of running the required level of AIX, and that has sufficient storage to meet the combined requirements of the programming prerequisites, MQSeries for AIX, the access methods, and the application programs:

- IBM RS/6000 POWERserver®
- IBM RS/6000 POWERstation
- IBM Scalable POWERparallel® systems
- Any other trademarked AIX systems, whether from IBM or other vendors, for example:
  - Bull DPX/20 (RISC)
  - Bull Escala (SMP)
  - Motorola
  - Zenith

An MQSeries client can run on any computer that is capable of running the client code and that has sufficient storage to meet the combined requirements of the programming prerequisites, the client code, access methods, and the application programs.

Software requirements

Software requirements are identical for server and client AIX environments unless otherwise stated.

- AIX Version, 4.2 or Version 4.3.x

  Note: Use level 4.2.1 or later if user data conversion of Greek, Cyrillic, Eastern European, Turkish, Japanese, or Korean language text longer than 2000 bytes is required.

Clients

Client code for AIX, DOS, HP-UX, OS/2 Warp, Sun Solaris, Windows NT, Windows 3.1, Windows 95, and Windows 98 workstations is distributed with the server code.

The MQSeries client for Java is also distributed with the server code; see the *MQSeries Clients* manual for more information.

Connectivity

- IBM eNetwork™ Communications Server for AIX, Version 5.0
- TCP (as part of the base operating system)

Note: MQSeries for AIX Version 5.1 supports the User Datagram Protocol (UDP), a part of the Internet suite of protocols, as an alternative to TCP. In the AIX environment, UDP is supplied as part of the operating system or TCP/IP suite you are using; you do not need to buy and install a separate UDP product. MQSeries support for UDP is available for MQSeries for AIX V5.1 in a service release. For information about the availability of the service release, contact your IBM representative or see the MQSeries Web site at:

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

Support for UDP in MQSeries for AIX V5.1 is for connection to MQSeries for Windows V2.0 only. UDP connectivity is introduced in MQSeries for Windows V2.0 in a service release, and is supported by SuperTCP only. The SuperTCP Suite is produced by Frontier Technologies Corporation.

You might decide to use UDP instead of TCP for your mobile radio network, where you need to reduce the traffic, and therefore the cost, on a packet radio data network.

Options

- Transaction processing managers (server only)
  - TXSeries for AIX, Version 4.2
  - BEA Tuxedo, Version 5.1, 6.1, or 6.4
- Databases (server only)
  - DB2 Universal Database™, Version 5.0
  - Oracle 7.3.2.1
  - Oracle 8.0.5
  - Sybase Adaptive Server, Version 11.5
  - Sybase Embedded SQL/C, Version 11.1
  - Sybase Open Client/C, Version 11.1.1
  - Sybase XA Server, Version 11.1.1 (SWR 7993)
- IBM Software Servers
  - Communications Server for AIX, Version 4.0
  - Database Server for AIX, Version 4.0
  - Directory Security Server for AIX, Version 4
  - Internet Connection Server for AIX, Version 4.1.1
  - Internet Connection Secure Server for AIX, Version 4.1
  - Transaction Server for AIX, Version 4.0
- IBM eNetwork LDAP Directory, Version 1.1.1
- DCE
  - IBM Directory and Security Server for AIX, Version 4
  - MQSeries DCE names and security modules are provided as part of MQSeries for AIX

Web functions

Web functions require Java, Version 1.1.1

- Web browsers
  - Internet Gateway: HTML 3.2 or later compatible browser
  - Java Client: HTML 3.2 plus JDK 1.1.1 enabled browser
- Web Servers for the Internet Gateway
  - Lotus® Domino™
  - Lotus Domino Go Webserver, Version 4.6
  - IBM Internet Connection Server, Version 4.1
  - Netscape FastTrack Server, Version 2.0
  - Netscape Enterprise Server, Version 2.0
  - Apache Web Server, Version 1.1.1
  - Microsoft Internet Information Server, Version 2.0

Languages and compilers

- IBM C and C++ compiler, Version 3.6
- IBM C for AIX, Version 4.4
- IBM C Set++ for AIX, Version 3.1
- Micro Focus COBOL compiler for UNIX, Version 4.0
- IBM COBOL Set for AIX, Version 1.1
- IBM PL/I set for AIX, Version 1.1
- IBM VisualAge Java Enterprise Edition for AIX, Version 2.0
- IBM VisualAge C++ Professional for AIX, Version 4.0

Delivery

MQSeries for AIX is supplied on CD-ROM. Two CD-ROMs are supplied, one containing the MQSeries for AIX server and client, and the other containing the other MQSeries clients shipped with MQSeries for AIX.

Installation

MQSeries for AIX is installed using either **Installation assistant**, **smit**, or **installp**.

The installation can be performed in approximately 5 minutes. Customization of the product is then required, the duration of this process being dependent on the individual requirements of the enterprise.

The *MQSeries for AIX V5.1 Quick Beginnings* manual contains instructions for installing MQSeries for AIX.

MQSeries for AS/400

IBM MQSeries for AS/400, Version 5 Release 1

Machine requirements

MQSeries for AS/400 V5R1 runs on any AS/400 processor capable of running the required level of OS/400 and that has enough processor storage to meet the combined requirements of the programming prerequisites, the access methods, and the application programs.

Software requirements

MQSeries for AS/400 V5R1 requires:

- OS/400 Version 4 Release 4 Modification 0

Clients

MQSeries for AS/400 supports client connections. Client code for AIX, DOS, HP-UX, OS/2 Warp, Sun Solaris, Windows NT, Windows 3.1, Windows 95, Windows 98, and Java environments is shipped with MQSeries for AS/400 V5R1.

There is no MQSeries for AS/400 client.

Connectivity

Connectivity can be through SNA LU 6.2 or TCP.

Options

- Transaction processing managers
  - CICS for AS/400 V4R4M0

    To use the MQI in application programs that operate under CICS you also require the program libraries for the programming language that you are using.

Languages and compilers

- ILE C for AS/400, V4R4M0
- ILE COBOL for AS/400, V4R4M0
- ILE RPG for AS/400, V4R4M0
- IBM ILE C++ for AS/400
- IBM VisualAge for C++ for AS/400
- Java application development: AS/400 Developer Kit for Java, 5769-JV1, and OS/400 Qshell Interpreter, 5769-SS1 Option 30

ILE run-time is part of OS/400.

Delivery

MQSeries for AS/400 V5R1 is supplied on CD-ROM.

Installation

MQSeries for AS/400 is installed using the OS/400 RSTLICPGM command. For installation instructions, refer to the *MQSeries for AS/400 V5.1 System Administration* book.

MQSeries for AT&T GIS UNIX

IBM MQSeries for AT&T GIS UNIX, Version 2 Release 2

Machine requirements

MQSeries for AT&T GIS UNIX runs on any AT&T GIS 34xx, 35xx, or 36xx system. A minimum of 20 MB of disk space is required.

Software requirements

- AT&T GIS UNIX SVR4 MP-RAS Version 2.0.3.01, or later Version 2.x
- NCR UNIX Release 3 or later Release 3.x

Clients

Client code for AT&T GIS UNIX, OS/2 Warp, DOS, and Windows 3.1 workstations is distributed with the server code.

Connectivity

The network protocols supported are SNA LU 6.2 and TCP.

- AT&T GIS SNA Services Version 2.06 or later Version 2, or equivalent NCR offering, to match hardware system
- TCP as part of base operating system

Options

- Transaction processing managers
  - BEA Tuxedo, Version 4.2.2

Languages and compilers

- NCR GIS High Performance C, Version 1.0b
- NCR C++ language system

Delivery

MQSeries for AT&T GIS UNIX is supplied on CD-ROM or QIC tape.

Installation

MQSeries for AT&T GIS UNIX is installed using the **pkgadd** command.

The installation can be performed in approximately 5 minutes. Customization of the product is then required, the duration of this process being dependent on the individual requirements of the enterprise.

The *MQSeries for AT&T GIS UNIX System Management Guide* contains instructions for installing this product.

MQSeries for Compaq (DIGITAL) OpenVMS

IBM MQSeries for Compaq (DIGITAL) OpenVMS Alpha, Version 2 Release 2.1.1
IBM MQSeries for Compaq (DIGITAL) OpenVMS VAX, Version 2 Release 2.1.1

Machine requirements

Any Digital VAX or Alpha machine with a minimum disk space of 20 MB

Software requirements

Software requirements are identical for server and client Compaq (DIGITAL) OpenVMS environments unless otherwise stated.

- Compaq (DIGITAL) OpenVMS Alpha Version 6.2, Version 7.1, or Version 7.2. (Version 7.2 requires system patch DEC-AXPVMS-VMS72_UPDATE-V0100-4.PCSI-DCX_AXPEXE from Compaq.)
- Compaq (DIGITAL) OpenVMS VAX Version 6.2, Version 7.1, or Version 7.2.

Clients

Client code for Compaq (DIGITAL) OpenVMS, OS/2 Warp, DOS, and Windows 3.1 workstations is distributed with the server code. The Windows 3.1 client can operate under Windows 3.1, Windows 95, or within the WIN-OS/2 environment under OS/2 Warp.

Connectivity

Network protocols supported are SNA LU 6.2, TCP, DECnet Phase IV, and DECnet Phase V.

Any communications hardware supporting SNA LU 6.2, TCP, or DECnet may be used.

For SNA connectivity, the SNA APPC LU6.2 software and license must be installed. It must have access to a suitably configured SNA gateway.

For TCP/IP connectivity, you need DEC TCP/IP Services for OpenVMS Version 4.0 or higher or Cisco Multinet Version 3.5 or higher.

Options

Distributed Computing Environment for OpenVMS, Version 1.3B

Languages and compilers

- DEC C, Version 5.0
- DEC C++, Version 5.0 (VAX) or Version 5.2 (AXP)
- DEC COBOL, Version 5.0 (VAX) or Version 2.2 (AXP)

Delivery

MQSeries for Compaq (DIGITAL) OpenVMS is supplied on CD-ROM for AXP and on CD-ROM and TK50 tape cartridge for VAX.

Installation

MQSeries for Compaq (DIGITAL) OpenVMS is installed using the OpenVMS VMSINSTAL utility. Installation takes approximately 15 minutes.

MQSeries client for DOS

This section summarizes the machine and software requirements for the MQSeries DOS client.

Client code for DOS workstations is distributed with the server code for all servers except OS/390, OS/400, Tandem NSK, and VSE/ESA.

Machine requirements

An MQSeries client can run under DOS on any personal computer that is capable of running the client code and that has sufficient storage to meet the combined requirements of the programming prerequisites, the client code, access methods, and the application programs.

Software requirements

- DOS 5.0

Connectivity

- Novell NetWare Client for DOS/Win, Version 1.2 or Version 2.5
- Novell LAN Workplace, Version 5.1
- FTP TCP for DOS, Version 5.0

Languages and compilers

- Microsoft C, Version 7.0
- Microsoft Visual C++, Version 1.5

MQSeries for HP-UX

IBM MQSeries for HP-UX, Version 5.1

Machine requirements

The MQSeries for HP-UX servers can run on any HP 9000 Family 700 or Family 800 or Stratus Continuum/400 machine. A minimum of 20 MB of disk space is required.

Software requirements

- HP-UX, Version 10.20 or 11.0

Clients

Client code for AIX, DOS, HP-UX, OS/2 Warp, Sun Solaris, Windows NT, Windows 3.1, Windows 95, and Windows 98 workstations is distributed with the server code.

The MQSeries client for Java is also distributed with the server code; see the *MQSeries Clients* manual for more information.

Connectivity

The network protocols supported are SNA LU 6.2 and TCP.

- HP SNAplusII
- TCP as part of base operating system

Options

- Transaction processing managers (server only)
  - TXSeries for HP-UX, Version 4.2
  - BEA Tuxedo, Version 5.1, 6.1, or 6.4
- Databases (server only)
  - DB2 Universal Database, Version 5.0
  - Oracle 7.3.2.3 (with patches 437448 and 441647)
  - Oracle 8.0.5
- IBM Software Servers
  - Internet Connection Server for HP-UX, Version 4.2.1
  - Internet Connection Secure Server for HP-UX, Version 4.2.1
- DCE
  - The HP DCE/9000 version appropriate for the level of the HP-UX operating system in use, providing that this is compatible with DCE Version 1.4.1.
  - MQSeries DCE names and security modules are provided as part of MQSeries for HP-UX.

Web functions

Web functions require Java, Version 1.1.1.

- Web browsers
  - Internet Gateway: HTML 3.2 or later compatible browser
  - Java Client: HTML 3.2 plus JDK 1.1.1 enabled browser
- Web Servers for the Internet Gateway
  - Lotus Domino
  - Lotus Domino Go Webserver, Version 4.6
  - IBM Internet Connection Server, Version 4.1
  - Netscape FastTrack Server, Version 2.1
  - Netscape Enterprise Server, Version 2.0
  - Apache Web Server, Version 1.1.1
  - Microsoft Internet Information Server, Version 2.0

Languages and compilers

- Bundled C compiler
- HP-UX ANSI C compiler
- C Softbench, Version 5.0
- IBM C compiler, Version 3.6
- IBM C++ compiler, Version 3.6
- HP C++, Version 3.1 for HP-UX Version 10.x
- HP-UX ANSI C++ compiler for HP-UX, Version 10 and Version 11
- Micro Focus COBOL compiler for UNIX, Version 4
- COBOL Softbench, Version 4.0
- HP DCE/9000 application development tools (with applicable patches)

Delivery

MQSeries for HP-UX is supplied on CD-ROM. Two CD-ROMs are supplied, one containing the MQSeries for HP-UX server and client, and the other containing the other MQSeries clients shipped with MQSeries for HP-UX.

Installation

MQSeries for HP-UX is installed using the **update** command.

The installation can be performed in approximately 5 minutes. Customization of the product is then required, the duration of this process being dependent on the individual requirements of the enterprise.

The *MQSeries for HP-UX V5.1 Quick Beginnings* manual contains instructions for installing this product.

MQSeries for OS/2 Warp

IBM MQSeries for OS/2 Warp, Version 5.1

Machine requirements

MQSeries for OS/2 Warp runs on any personal computer that is capable of running the required level of OS/2 Warp, and which has sufficient RAM and disk storage to meet the combined requirements of the programming prerequisites, MQSeries for OS/2 Warp, the access methods, and the application programs. The system unit must have a CD-ROM device.

An MQSeries client can run on any computer that is capable of running the client code and that has sufficient storage to meet the combined requirements of the programming prerequisites, the client code, access methods, and the application programs.

Software requirements

Software requirements are identical for server and client OS/2 Warp environments unless otherwise stated.

- OS/2 Warp, Version 4.0
- OS/2 Warp Server, Version 4.0
- OS/2 Warp Server Advanced SMP feature, Version 4.0
- OS/2 Workspace-On-Demand
- OS/2 e-Business Server

Clients

Client code for AIX, DOS, HP-UX, OS/2 Warp, Sun Solaris, Windows NT, Windows 3.1, Windows 95, and Windows 98 workstations is distributed with the server code.

The MQSeries client for Java is also distributed with the server code; see the *MQSeries Clients* manual for more information.

Connectivity

- IBM eNetwork Communications Server for OS/2 Warp, Version 5.0
- Novell NetWare Client for OS/2, Version 1.20
- TCP/IP for OS/2, Version 3.5, base kit plus NetBIOS kit
- intraNetWare Client for OS/2, Version 2.12
- NetWare for OS/2, Version 4.11

Options

- Transaction processing managers (server only)
  - CICS Transaction Server for OS/2, Version 4.1
- Databases (server only)
  - DB2 Universal Database, Version 5.0
- IBM Software Servers
  - Database Server for OS/2 Warp, Version 4.0
  - Directory Security Server for OS/2 Warp, Version 4.0
  - Internet Connection Server for OS/2 Warp, Version 4.2.1
  - Internet Connection Secure Server for OS/2 Warp, Version 4.2.1
  - Transaction Server for OS/2, Version 4.0

- DCE
  - IBM Directory and Security Server for OS/2 Warp, Version 4.0.
  - MQSeries DCE names and security modules are provided as part of MQSeries for OS/2 Warp.

Web functions

Web functions require Java, Version 1.1.1.

- Web browsers
  - Internet Gateway: HTML 3.2 or later compatible browser
  - Java Client: HTML 3.2 plus JDK 1.1.1 enabled browser
- Web Servers for the Internet Gateway
  - Lotus Domino
  - Lotus Domino Go Webserver, Version 4.6
  - IBM Internet Connection Server, Version 4.1
  - Netscape FastTrack Server, Version 2.0
  - Netscape Enterprise Server, Version 2.0
  - Apache Web Server, Version 1.1.1
  - Microsoft Internet Information Server, Version 2.0

Languages and compilers

- IBM C compiler, Version 3.6
- IBM C++ compiler, Version 3.6
- Borland C++ Compiler, Version 2.0 or Version 5.02 (C bindings)
- IBM VisualAge for C++ for OS/2, Version 3.0
- Micro Focus COBOL, Version 4
- IBM VisualAge for COBOL for OS/2, Version 1.1 or Version 2.2
- IBM PL/I for OS/2, Version 1.2
- IBM VisualAge for PL/I for OS/2
- IBM VisualAge Java Enterprise Edition for OS/2, Version 2.0
- IBM VisualAge Java Professional Edition for OS/2, Version 2.0

Delivery

MQSeries for OS/2 Warp is supplied on CD-ROM. Two CD-ROMs are supplied, one containing the MQSeries for OS/2 Warp server and client, and the other containing the other MQSeries clients shipped with MQSeries for OS/2 Warp.

Installation

MQSeries for OS/2 Warp is installed using Software Installer/2. The installation can be performed in approximately 15 minutes. Customization of the product is then required, the duration of this process being dependent on the individual requirements of the enterprise.

The *MQSeries for OS/2 Warp V5.1 Quick Beginnings* manual contains instructions for installing MQSeries for OS/2 Warp. It also gives information about using Configuration, Installation, and Distribution (CID), and NetView Distribution Manager/2 (DM/2) to install MQSeries for OS/2 Warp.

MQSeries for OS/390

IBM MQSeries for OS/390, Version 2 Release 1

Machine requirements

MQSeries for OS/390 runs on any IBM System/390 processor that is capable of running the required level of OS/390 and that has enough storage to meet the combined requirements of the programming prerequisites, MQSeries for OS/390, the access methods, and the application programs.

Software requirements

This section lists the software requirements for Version 2.1 of MQSeries for OS/390. We recommend that you use one of the packaged offerings of OS/390 from IBM (for example, ServerPac). This includes most of the products that you will need to run MQSeries on OS/390, and the integrated products have been verified by IBM.

The list of elements included in OS/390 and the recommended levels of nonexclusive elements are described in *OS/390 Planning For Installation* (GC28-1726).

MQSeries for OS/390 Version 2.1 requires OS/390 Version 2.4 or later, together with the products included in OS/390. These include:

- C/C++
- DFSMS™/dfp
- HL/ASM V1.2
- ISPF V4.2.1
- JES
- Language Environment® (previously known as LE/370)
- Security Server (previously known as RACF)
- SMP/E
- TCP/IP
- TSO/E
- UNIX Services (previously known as OpenEdition®)
- VTAM®

The following list gives the minimum levels required for the optional products that are not included in OS/390. You might not need all of these products.

- CICS Transaction Server for OS/390, Version 4.1
- COBOL, Version 2.1, or VS COBOL 2, Release 4
- HL/ASM, Version 1.2
- Internet Connection Secure Server, Version 2.2
- IMS, Version 5.1
- ISPF, Version 4.2.1
- PL/I for OS/390, Version 1.1, or OS PL/I, Version 2.3

Additional requirements for some features

Some of the features of MQSeries for OS/390 V2.1 have additional requirements. These are listed below.

TCP IUCV interface

This can be used only on OS/390, Version 2.4. For later versions of OS/390, use the OpenEdition Sockets interface to TCP.

MQSeries Workflow

- OS/390, Version 2.6
- MQSeries Workflow, Version 3.1

Non-IBM products

If you use Interlink TCPaccess in place of IBM's TCP/IP, the recommended minimum level is Version 4.1.

Clients

For MQSeries for OS/390 to support clients you need to install the client/server support code that is provided by the Client Attachment Feature of MQSeries for OS/390. However, you can administer clients without this feature.

Delivery

MQSeries for OS/390 is supplied on either 6250 tape or 3480 cartridge. (It is also available on 4-mm DAT tape for PC/390.) One tape or cartridge contains the product code together with four language features – U.S. English (mixed case), U.S. English (uppercase), Japanese, and Simplified Chinese – and the optional CICS mover facility. The optional Client Attachment and Internet Gateway features are both supplied on separate tapes or cartridges.

Installation

MQSeries for OS/390 is installed with SMP/E using the receive-apply-accept approach. JCL is provided on the tape to assist with this process.

The installation can be performed in approximately two hours. Customization of the product is then required, the duration of this process being dependent on the individual requirements of the enterprise.

The *MQSeries for OS/390 Program Directory* contains instructions for installing MQSeries for OS/390.

MQSeries for SINIX and DC/OSx

IBM MQSeries for SINIX and DC/OSx, Version 2.2

Machine requirements

SINIX: RM200, RM300, RM400, RM600 systems with minimum disk space of 30 MB. If DynaText books are installed, a minimum of 50 MB of system disk space is needed.

DC/OSx: MIServer, Nile systems with minimum disk space of 30 MB.

Software requirements

Software requirements are identical for server and client environments unless otherwise stated.

- SINIX operating system, for RM200, RM300, or RM400: SINIX-N Version 5.42C10 or Reliant Version 5.43
- SINIX operating system, for RM600: SINIX-Y Version 5.42A40 or Reliant Version 5.43
- DC/OSx operating system Version 1.1-cd079

Clients

Client code for SINIX and DC/OSx, OS/2 Warp, DOS, and Windows 3.1 workstations is distributed with the server code.

The Windows 3.1 client can operate under Windows 3.1, Windows 95 or within the WIN-OS/2 environment under OS/2 Warp.

Client software provides a remote interface to a LAN server. It can reside at the server or at a file server and be copied dynamically to the client for use, or it can reside on the client's disk space.

Client support does not result in distributed coordination of units of work.

Connectivity

The network protocols supported are SNA LU 6.2 and TCP.

- SINIX: SNA
  TRANSIT-SERVER 3.4 (SNA Communication Server Version)
  TRANSIT-CLIENT 3.4 (SNA Comm. Client / Local Functions)
  TRANSIT-CPIC 3.4 (SNA LU 6.2 Communication and CPI-C)
- SINIX: OpenNet TCP/IP (shipped with base SINIX operating system)
- DC/OSx: TCP/IP, Version 1.0
- DC/OSx: SNA requires LU 6.2 SW, Version 1.3
  - To support the ISC-2 (Intelligent Synchronous Controller) serial line
    - Comm Services V 1.2 and ISC with SNA engine V 3.1
  - To support the ILC-T (Intelligent LAN Controller, Token ring) interface
    - Comm Services V 1.2 and Token Ring Mac interface V 1.3
  - To support SNA on the ESCON® IBM Channel link
    XVI/ESCON Driver 1.0

Options

- Transaction processing managers
  - SINIX: Encina, Version 1.1A00
  - SINIX: CICS for Siemens Nixdorf SINIX, Version 2.1.1
  - SINIX: Novell Tuxedo, Version 5.0
  - DC/OSx: Novell Tuxedo, Version 5.0
- DCE
  - SINIX: Version DCE-MI V1.03B00

Note: DCE cannot be invoked from installable services or from user exits on the SINIX platform. However, stand-alone DCE programs can invoke the MQI.

Languages and compilers

- SINIX: C compiler (C-DS, MIPS), Version 1.1
- DC/OSx: C 4.0 compiler, Version 4.0.1
- SINIX: Micro Focus COBOL, Version 3.2
- DC/OSx: Micro Focus COBOL, Version 3.2

Delivery

- For SINIX, MQSeries is supplied on CD-ROM.
- For DC/OSx, MQSeries is supplied on QIC-320 cartridge tape.

Documentation browsing

On SINIX, to view the online documentation, DynaText is required. Either use SINIX/Windows Version 2 or later, which provides a DynaText viewer, or use the SINIX online documentation package (separate product), which also provides a DynaText viewer.

Installation

MQSeries for SINIX is installed using the **sysadm** command. MQSeries for DC/OSx is installed using the **pkgadd** command.

MQSeries for Sun Solaris

IBM MQSeries for Sun Solaris, Version 5.1

Machine requirements

Only the Sun SPARC and Sun UltraSPARC hardware is supported. A minimum of 25 MB of system disk space is required.

Software requirements

Software requirements are identical for server and client on Solaris environments unless otherwise stated.

- Sun Solaris, Version 2.6 (with patches 105181-12, 105210-19, 105490-07, 105568-13, and 105591-05) or Version 7.

Clients

Client code for AIX, DOS, HP-UX, OS/2 Warp, Sun Solaris, Windows NT, Windows 3.1, Windows 95, and Windows 98 workstations is distributed with the server code.

The MQSeries client for Java is also distributed with the server code; see the *MQSeries Clients* manual for more information.

Connectivity

The network protocols supported are SNA LU 6.2 and TCP.

- SunLink SNA Peer-to-Peer, Version 9.1
- If a token ring is to be used, SunLink Token Ring Interface /SBus, Version 3.0.2 (with patch 102463) or Sun TRI/P Adapter, Version 1.0
- TCP as part of the base operating system

Options

- Transaction processing managers (server only)
  - TXSeries for Sun Solaris, Version 4.2
  - BEA Tuxedo, Version 5.1, 6.1, or 6.4
- Databases (server only)
  - DB2 Universal Database, Version 5.0
  - Oracle 7.3.2.3
  - Oracle 8.0.5
  - Sybase Adaptive Server, Version 11.5
  - Sybase Embedded SQL/C, Version 11.1
  - Sybase Open Client/C, Version 11.1.1
  - Sybase XA Server, Version 11.1.1 (EBF7589)
- IBM Software Servers
  - Internet Connection Server for Sun Solaris, Version 4.1
  - Internet Connection Secure Server for Sun Solaris, Version 4.1
- DCE
  - Transarc DCE, Version 1.1.
  - MQSeries DCE names and security modules are provided as part of MQSeries for Sun Solaris.

Web functions

Web functions require Java, Version 1.1.3 plus the native thread pack.

- Web browsers
  - Internet Gateway: HTML 3.2 or later compatible browser
  - Java Client: HTML 3.2 plus JDK 1.1.1 enabled browser
- Web Servers for the Internet Gateway
  - Lotus Domino
  - Lotus Domino Go Webserver, Version 4.6
  - IBM Internet Connection Server, Version 4.1
  - Netscape FastTrack Server, Version 2.0
  - Netscape Enterprise Server, Version 2.0
  - Apache Web Server, Version 1.1.1
  - Microsoft Internet Information Server, Version 2.0

Languages and compilers

- SunWorkShop C Compiler, Version 4.2
- SunWorkShop C++ Compiler, Version 4.2
- Micro Focus COBOL for UNIX, Version 4.0

Delivery

MQSeries for Sun Solaris is supplied on CD-ROM. Two CD-ROMs are supplied, one containing the MQSeries for Sun Solaris server and client, and the other containing the other MQSeries clients shipped with MQSeries for Sun Solaris.

Installation

MQSeries for Sun Solaris is installed using the **pkgadd** command or the **sysadm** command. The *MQSeries for Sun Solaris V5.1 Quick Beginnings* manual contains instructions for installing this product.

MQSeries for Tandem NSK

IBM MQSeries for Tandem NSK, Version 2 Release 2.0.1

Machine requirements

Minimum hardware requirements are:

- Any of the Tandem NSK range of machines supported by Tandem NSK D3x, D4x, G02, or later G0x
- Specific hardware in support of user-selected network transport protocols

You are also recommended to have one or more mirrored data disks with specified space requirements for TMF audit space and the MQSeries database.

Software requirements

- Tandem NSK D3x, D4x, G02, or later G0x operating systems, including TM/MP (TMF), ENSCRIBE, and EMS
- TS/MP (PATHWAY) to match the operating system
- SCF for configuration, command, and control of TCP and SNA network transports

Transaction logging is maintained with the Tandem TM/MP (TMF) product.

Connectivity

For SNA connectivity, one of:

- SNAX/APC and SNAX/XF
- SNAX/APN
- Insession ICE

to match the operating system

For TCP connectivity:

- TCP to match the operating system

Clients

MQSeries for Tandem NSK supports, but does not ship, MQSeries clients.

Languages and compilers

Supported languages for application development:

- TAL
- C
- COBOL-85

Delivery

MQSeries for Tandem NSK is supplied on a 3480 cartridge.

Installation

The *MQSeries for Tandem NonStop Kernel System Management Guide* contains instructions for installing this product.

MQSeries client for VM/ESA

This section summarizes the machine and software requirements for the MQSeries client for VM/ESA.

Machine requirements

An MQSeries client can run under VM/ESA on any of the following machines:

- S/390 Parallel Enterprise Server™ - Generation 3
- S/390 Multiprise® 2000
- ES/9000® Processors

Software requirements

The following are prerequisites for MQSeries applications running on a VM/ESA client.

- VM/ESA, Version 2.3
- LE/370, Release 1.6
- TCP/IP, Release 2.0, or VTAM LU 6.2

Languages and compilers

- IBM C for VM, Release 3.1
- IBM VS COBOL II
- IBM OS/PL/I, Release 2.3
- IBM VM/ESA REXX/VM
- IBM Assembler

Delivery

The MQSeries client for VM/ESA is shipped with the VM/ESA product. It is preloaded with the installation of CMS, and can be accessed on the `MAINT 193` disk.

MQSeries for VSE/ESA

IBM MQSeries for VSE/ESA, Version 2 Release 1

Machine requirements

- Any IBM System 370 or 390 machine
- Any communications hardware supporting TCP or SNA LU 6.2

Software requirements

- VSE/ESA Version 2.3 or later 2.x

Clients

MQSeries for VSE/ESA can act as a server to an MQSeries client, but does not ship or configure clients.

Connectivity

- For SNA, VTAM for VSE/ESA, Version 4.2
- For TCP, TCP/IP for VSE/ESA, Version 1.3, with appropriate PTFs

Options

- CICS/VSE, Version 2.3
- LE/VSE runtime library, Version 1.4

Languages and compilers

- IBM C for VSE/ESA, Version 1.1
- IBM COBOL for VSE/ESA, Version 1.1
- IBM PL/I for VSE/ESA, Version 1.1

Delivery

MQSeries for VSE/ESA is supplied on 3480 cartridge, 6250 tape, or 4mm DAT tape.

Installation

The MQSeries for VSE/ESA tape is in standard IBM MSHP format.

The *MQSeries for VSE/ESA System Management Guide* contains instructions for installing this product.

MQSeries for Windows Version 2.0

IBM MQSeries for Windows, Version 2.0

Machine requirements

MQSeries for Windows V2.0 is a 16-bit product: it runs on computers that run Windows 3.1, and runs in 16-bit compatibility mode on Windows 95. Table 22 suggests two configurations: one for running applications and the other for developing applications.

Note: These recommendations are for guidance only. They do not take into account the effects of any other software that might be running on the system at the same time.

Table 22. Suggested hardware configurations for MQSeries for Windows, Version 2.0

Configuration	Processor	RAM	Hard disk
For running applications	386 16 MHz	4–8 MB	3.9 MB available
For developing applications	486 66 MHz or better	8–16 MB	5 MB available
Note: The specification for developing applications does not include hardware requirements for other development tools (for example, compilers).			

Software requirements

This section describes the software you require before you can use MQSeries for Windows, V2.0. This depends on whether you want to run MQSeries applications on MQSeries for Windows, or develop your own applications for it.

For running MQSeries applications

For running applications on MQSeries for Windows, V2.0, you need the following software:

- MS-DOS or PC DOS, Version 6.2
- Microsoft Windows 3.1, Windows 95, Windows for Workgroups 3.11, or Win-OS/2 on OS/2, Version 3.0 (Warp)

  MQSeries for Windows V2.0 runs in 16-bit compatibility mode on Windows 95.

Connectivity

- TCP for the operating system you are using:
  - For Microsoft Windows 3.1, you need IBM TCP/IP for DOS, Version 2.1.1 with CSD 2.1.1.4.
  - For Microsoft Windows 95, use the version of TCP supplied with Windows 95.
  - For Microsoft Windows for Workgroups 3.11, you need IBM TCP/IP for DOS, Version 2.1.1 with CSD 2.1.1.4.
  - For Win-OS/2, you need IBM TCP/IP for OS/2, Version 2 or Version 3.0.

Note: MQSeries for Windows V2.0 supports, via a service release, the User Datagram Protocol (UDP), a part of the Internet suite of protocols, as an alternative to TCP. You might decide to use UDP instead of TCP for your mobile radio network, where you need to reduce the traffic, and therefore the cost, on a packet radio data network.

UDP is supplied as part of the operating system or TCP/IP suite you are using; you do not need to buy and install a separate UDP product.

Languages and compilers

To develop and test MQSeries applications that run on Windows, in addition to the software listed in "For running MQSeries applications" on page 250, you need only the compiler for the programming language you will use.

For 16-bit C
Microsoft Visual C++, Version 1.5

For 32-bit C
Microsoft Visual C++, Version 2.0

For 16-bit BASIC
Microsoft Visual Basic, Version 3.0 or Microsoft Visual Basic, Version 4.0

For 32-bit BASIC
Microsoft Visual Basic, Version 4.0

MQSeries for Windows, Version 2.0 runs in 16-bit compatibility mode on Windows 95, but you can write 32-bit MQSeries for Windows applications.

Delivery

MQSeries for Windows, Version 2.0 is supplied on diskettes. It is enabled for Configuration, Installation, and Distribution (CID) so you can put the installation files on a LAN server for easier access.

Installation

The *MQSeries for Windows V2.0 User's Guide* contains instructions for installing this product.

MQSeries for Windows Version 2.1

IBM MQSeries for Windows, Version 2.1

Machine requirements

MQSeries for Windows V2.1 is a 32-bit product. It runs on computers that run Windows 95, Windows 98, or Windows NT Version 4.0. Table 23 suggests two configurations, one for running applications and the other for developing applications.

Note: These recommendations are for guidance only. They do not take into account the effects of any other software that might be running on the system at the same time.

Table 23. Suggested hardware configurations for MQSeries for Windows, Version 2.1

Configuration	Processor	RAM	Hard disk
For running applications	386DX or better	At least 4 MB	At least 3.5 MB available
For developing applications	486 66 MHz or better	At least 8 MB	At least 5 MB available
Note: The specification for developing applications does not include hardware requirements for other development tools (for example, compilers).			

Software requirements

This section describes the software you require before you can use MQSeries for Windows, Version 2.1. This depends on whether you want to run MQSeries applications on MQSeries for Windows, or develop your own applications for it.

For running MQSeries applications

For running applications on MQSeries for Windows, Version 2.1, you need the following software:

- Microsoft Windows 95, Windows 98, or Windows NT Version 4.0

Connectivity

- TCP for the operating system you are using.

Languages and compilers

To develop and test MQSeries applications that run on Windows, in addition to the software listed in “For running MQSeries applications”, you need only the compiler for the programming language you will use.

- Microsoft Visual C++, Version 4.0
- Borland C
- Microsoft Visual Basic, Version 4.0

Delivery

MQSeries for Windows, Version 2.1 is supplied on CD-ROM or diskettes. It is enabled for remote (or silent) installation so you can put the installation files on a LAN server for easier access.

Installation

The *MQSeries for Windows V2.1 User's Guide* contains instructions for installing this product.

MQSeries for Windows NT

IBM MQSeries for Windows NT, Version 5.1

Machine requirements

MQSeries for Windows NT can run on any Intel 486 (or above) processor-based IBM PC machine or compatible.

Software requirements

- Microsoft Windows NT, Version 4.0 (with Service Pack 3 and Service Pack 4)
- Microsoft Windows 2000, with CSD03 applied to MQSeries for Windows NT V5.1

Clients

Client code for AIX, DOS, HP-UX, OS/2 Warp, Sun Solaris, Windows NT, Windows 3.1, Windows 95, and Windows 98 workstations is distributed with the server code.

The MQSeries client for Java is also distributed with the server code; see the *MQSeries Clients* manual for more information.

Connectivity

The network protocols supported are SNA LU 6.2, TCP, NetBIOS, and SPX.

- SNA LU 6.2
  - Attachmate EXTRA! Personal Client, Version 6.1 and 6.2
  - IBM Communications Server for Windows NT, Version 5.0 and Version 6.0
  - Microsoft SNA Server, Version 2.11, Version 3, and Version 4.0
- For TCP use the TCP facilities within Windows NT
- For NetBIOS use the NetBIOS facilities within Windows NT
- For SPX use the SPX facilities within Windows NT

Options

- Transaction processing managers (server only)
  - TXSeries for Windows NT, Version 4.2
  - BEA TUXEDO, Version 5.1, 6.1, or 6.4
- Databases
  - DB2 Universal Database, Version 5.0
  - Sybase Adaptive Server Enterprise, Version 11.5
  - Sybase Embedded SQL/C, Version 11.1.0
  - Sybase Open Client (ctlib and dblib), Version 11.1.1 (with latest support level)
  - Sybase XA Server, Version 11.1.1 (with latest support level)
- IBM Software Servers
  - Communications Server for Windows NT, Version 5.0
  - Database Server for Windows NT, Version 4.0
  - Directory Security Server for Windows NT, Version 5.0
  - Internet Connection Server for Windows NT, Version 4.2.1
  - Internet Connection Secure Server for Windows NT, Version 4.2.1
  - Transaction Server for Windows NT, Version 4.0
- DCE
  - IBM DCE, Version 1.1.
  - MQSeries DCE security modules are provided as part of MQSeries for Windows NT.
- Microsoft Management Console, Version 1.1

- Active Directory Service Interfaces, Version 2.0
- HTMLHelp, Version 1.1

Web functions

Web functions require Java, Version 1.1.1.

- Web browsers
  - Internet Gateway: HTML 3.2 or later compatible browser
  - Java Client: HTML 3.2 plus JDK 1.1.1 enabled browser
- Web administration
  - Netscape Navigator, Version 4.04 with the Java AWT upgrade
  - Internet Explorer, Version 4.01 SP1
- Web Servers for the Internet Gateway
  - Lotus Domino
  - Lotus Domino Go Webserver, Version 4.6
  - IBM Internet Connection Server, Version 4.1
  - Netscape FastTrack Server, Version 2.0
  - Netscape Enterprise Server, Version 2.0
  - Apache Web Server, Version 1.1.1
  - Microsoft Internet Information Server, Version 2.0

Languages and compilers

- IBM C compiler, Version 3.6
- IBM C++ compiler, Version 3.6
- Microsoft Visual C++ for Windows 95 and NT, Version 4.0, Version 5.0, and Version 6.0
- IBM VisualAge for C++ for Windows, Version 3.5
- IBM VisualAge C++ Professional, Version 4.0
- IBM VisualAge COBOL for Windows NT, Version 2.1
- IBM VisualAge COBOL Enterprise, Version 2.2
- IBM PL/I for Windows, Version 1.2
- IBM VisualAge for PL/I for Windows
- IBM VisualAge PL/I Enterprise, Version 2.1
- IBM VisualAge e-business for Windows, Version 1.0.1
- IBM VisualAge for Java Enterprise, Version 2.0
- IBM VisualAge for Java Professional, Version 2.0
- Visual Basic for Windows, Version 4.0 and 5.0

Delivery

MQSeries for Windows NT is supplied on CD-ROM. Two CD-ROMs are supplied, one containing the MQSeries for Windows NT server and client, and the other containing the other MQSeries clients shipped with MQSeries for Windows NT.

Installation

MQSeries for Windows NT is installed using the **setup** utility.

The installation can be performed in approximately 15 minutes. The actual time taken depends on several factors, including the following:

- The speed of the machine
- Which functions you are installing
- Where you are installing from

Customization of the product is then required, the duration of this process being dependent on the individual requirements of the enterprise.

MQSeries for Windows NT

The *MQSeries for Windows NT V5.1 Quick Beginnings* manual contains instructions for installing this product.

MQSeries client for Windows 3.1

This section summarizes the machine and software requirements for the Windows 3.1 client.

Client code for Windows 3.1 workstations is distributed with the server code for all servers except OS/390, OS/400, Tandem NSK, and VSE/ESA.

Machine requirements

An MQSeries client can run under Windows 3.1 on any personal computer that is capable of running the client code and that has sufficient storage to meet the combined requirements of the programming prerequisites, the client code, access methods, and the application programs.

Software requirements

- Microsoft Windows 3.1
- Microsoft Windows 95 in 16-bit mode
- WIN-OS2 environment within OS/2

Connectivity

- IBM TCP/IP for OS/2 base kit (part of the base operating system) and DOS/Windows access kit (allows clients access to TCP via programs that run under WIN-OS2)
- Novell NetWare Client for DOS/Win, Version 1.2
- Novell NetWare Client for OS/2, Version 2.1 (allows clients to access SPX via programs that run under WIN-OS2)
- Other TCPs:
  - PC-NSFPro
  - Sunsoft
  - OnNet SDK for Windows

Languages and compilers

- Microsoft C, Version 7.0
- Microsoft Visual C++, Version 1.5

MQSeries client for Windows 95 and Windows 98

This section summarizes the machine and software requirements for the MQSeries client for Windows 95 and Windows 98.

Client code for Windows 95 and Windows 98 workstations is distributed with the server code for all servers except OS/390, OS/400, Tandem NSK, and VSE/ESA.

Machine requirements

An MQSeries client can run under Windows 95 or Windows 98 on any personal computer that is capable of running the client code and that has sufficient storage to meet the combined requirements of the programming prerequisites, the client code, access methods, and the application programs.

Software requirements

- Microsoft Windows 95
- Microsoft Windows 98

Connectivity

TCP, SPX, and NetBIOS are all provided in the operating system.

Languages and compilers

- IBM VisualAge C++, Version 3.5
- Microsoft Visual C++, Version 4.0
- Micro Focus COBOL Workbench, Version 4.0

MQSeries link for R/3

This section tells you about the hardware and software you need to run MQSeries link for R/3.

Hardware requirements

To install and run this product, you need approximately 5 MB of available hard disk space. There are no additional hardware requirements except for those listed for MQSeries and R/3 on the platform you are using.

Software requirements

To run the R/3 link, you must have installed the following:

- At least one R/3 system, Version 3.0E or later
- The MQSeries server product for the chosen platform
- The relevant MQSeries link for R/3 product for the chosen platform
- On Windows NT, the **librfc32.dll** file that is supplied with the SAPGUI component of R/3

Restrictions

The amount of R/3 IDoc data that can be sent in a single message is limited by the maximum MQSeries message size. With MQSeries Version 5 the maximum message size is 100 MB and with earlier versions of MQSeries the maximum message size is 4 MB. Because the MQSeries link for R/3 header files are part of the space that is assigned to MQSeries messages, the actual space available for R/3 transaction data is less than the full 100 MB (or 4MB). R/3 transaction data can consist of one IDoc or batched IDocs.

SAP AG recommends a maximum of 2 MB for each IDoc.

Supported platforms

MQSeries link for R/3 is available for the following release levels of these platforms:

- AIX V4.1.4
- OS/400 V4R2
- DIGITAL UNIX (Compaq Tru64 UNIX) 4.0.D
- HP-UX V10.10 and V11
- Sun Solaris V2.5.1
- Windows NT V3.5.1 with service pack 5 installed

MQSeries platforms

Using R/3 link, you can connect your R/3 system to other platforms that run MQSeries.

MQSeries PRPQ (limited availability) products

The MQSeries PRPQ (programming request for price quotation) server products are:

- MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX), Version 2.2.1
- MQSeries for Solaris on Intel, Version 5.0
- MQSeries for Windows NT on DIGITAL Alpha (AXP), Version 2.0.1

For further information about these PRPQ products, contact your IBM marketing representative or see the MQSeries home page at:

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

MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX)

IBM MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX), Version 2 Release 2.1

Machine requirements

The MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX), Version 2 Release 2.1 server code runs on any DIGITAL Alpha machine capable of running DIGITAL UNIX (Compaq Tru64 UNIX) 4.0.D. The minimum system disk space required is 20 MB.

Operating system

Requirements are identical for server and client DIGITAL UNIX environments:

- DIGITAL UNIX (Compaq Tru64 UNIX) V4.0.D

Clients

Client code for DIGITAL UNIX (Compaq Tru64 UNIX) is distributed with the server code.

Connectivity

Any communications hardware supporting TCP in the DIGITAL UNIX (Compaq Tru64 UNIX) environment may be used.

Languages and compilers

- DEC C for DIGITAL UNIX (Compaq Tru64 UNIX), V5.2

Delivery

MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX) is supplied on CD-ROM.

Installation

MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX) is installed using the `setld` command. Installation takes approximately 5 minutes.

MQSeries for Solaris on Intel

IBM MQSeries for Solaris on Intel, Version 5 Release 0.

Machine requirements

For information about the Intel hardware on which you can install the MQSeries for Solaris server and client, and about which Token Ring cards are supported, see the Compatibility List accessible from the following Web site:

```
http://access1.sun.com
```

From the Compatibility List, see the section titled *Solaris 2.5/2.5.1 x.86 Hardware Compatibility List*.

Operating system

Software requirements are identical for server and client in Solaris environments:

- Sun Solaris, Version 2.5.1

Clients

The MQSeries client for Solaris on Intel and the MQSeries client for Java are distributed with the server code.

If you install the MQSeries client for Java and the MQSeries bindings for Java, you need to apply the following patches to Sun Solaris Version 2.5.1:

- 103641-17
- 104241-05

In addition, the following versions of Java run-time code must be installed on your machine:

- MQSeries client for Java: Java Version 1.1.7
- MQSeries bindings for Java: Java Version 1.1.7 (Native threads), which is available from:

  ```
  http://www.sun.com/solaris/java/index.html
  ```

Connectivity

MQSeries for Solaris on Intel supports the TCP communications protocol (supplied with the Sun Solaris operating system). You need:

- Any communications hardware that supports TCP
- If using Token Ring hardware, SunLink Token Ring Interface/PCI Bus

Options

- Databases (server only)
  - Oracle8 Server Release 8.0.5. You must apply the Oracle patch for bug 798509 for XA coordination.
  - Oracle 7.3.2.3 and MQSeries can coexist on a single machine. However, they do not interoperate in an XA-compliant way.

Languages and compilers

- SunWorkShop C Compiler, Version 4.2 (in the Sun Visual Workshop Version 3.0)

Delivery

MQSeries for Solaris on Intel is supplied on CD-ROM.

Installation

MQSeries for Solaris on Intel is installed using the **pkgadd** command. The *MQSeries for Sun Solaris Version 5.0 Quick Beginnings* book contains instructions for installing this product.

MQSeries for Windows NT on DIGITAL Alpha

IBM MQSeries for Windows NT on DIGITAL Alpha (AXP), Version 2 Release 0.1

Machine requirements

The MQSeries for Windows NT on DIGITAL Alpha (AXP) server runs on any DIGITAL Alpha (AXP) machine with at least 8 MB of free system disk space and approximately 20 MB of working space.

Operating system

- Microsoft Windows NT V4.0

Clients

The MQSeries client for Windows NT on DIGITAL Alpha (AXP) is available as a SupportPac from the MQSeries Web site at:

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

Connectivity

MQSeries for Windows NT on DIGITAL Alpha (AXP) supports the SNA and TCP communications protocols. Requirements are:

- Any communications hardware that supports the selected protocol
- For SNA, Microsoft SNA server V2.11

Languages and compilers

- Micro Focus Object COBOL for Windows NT, V4

Delivery

MQSeries for Windows NT on DIGITAL Alpha (AXP) is supplied on CD-ROM.

Installation

For installation instructions, see the documentation included in the product package.

Part 12. Appendixes

Appendix. Notices

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

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

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

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Trademarks

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

AIX	AS/400	BookManager
CICS	CICS/VSE	DB2
DB2 Universal Database	DFSMS	eNetwork
ESCON	ES/9000	HACMP/6000
IBM	IMS	IMS/ESA
Integrated Language Environment	Language Environment	MQ
MQSeries	Multiprise	MVS/ESA
Open Blueprint	OpenEdition	OS/2
OS/390	OS/400	POWERparallel
POWERserver	RACF	RISC System/6000
RMF	RS/6000	SupportPac
System/390	S/390	S/390 Parallel Enterprise Server
TXSeries	VisualAge	VM/ESA
VSE/ESA	VTAM	WIN-OS/2

Lotus and Domino are trademarks of Lotus Development Corporation in the United States, other countries, or both.

NetView is a trademark of Tivoli Systems Inc. in the United States, other countries, or both.

Intel is a trademark of Intel Corporation in the United States, other countries, or both.

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

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

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

Other company, product, or service names, may be the 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

active log. See *recovery log*.

adapter. An interface between MQSeries for OS/390 and TSO, IMS, CICS, or batch address spaces. An adapter is an attachment facility that enables applications to access MQSeries services.

address space. The area of virtual storage available for a particular job.

administrator commands. MQSeries commands used to manage MQSeries objects, such as queues, processes, and namelists.

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 environment. The software facilities that are accessible by an application program. On the OS/390 platform, CICS and IMS are examples of application environments.

archive log. See *recovery log*.

ARM. Automatic Restart Management

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 service. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, a service that provides authority checking of commands and MQI calls for the user identifier associated with the command or call.

authorized program analysis report (APAR). A report of a problem caused by a suspected defect in a current, unaltered release of a program.

Automatic Restart Management (ARM). An OS/390 recovery function that can improve the availability of specific batch jobs or started tasks, and therefore result in faster resumption of productive work.

B

backout. 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*.

bag. See *data bag*.

bootstrap data set (BSDS). A VSAM data set that contains:

- An inventory of all active and archived log data sets known to MQSeries for OS/390
- A wrap-around inventory of all recent MQSeries for OS/390 activity

The BSDS is required if the MQSeries for OS/390 subsystem has to be restarted.

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.

BSDS. Bootstrap data set.

buffer pool. An area of main storage used for MQSeries for OS/390 queues, messages, and object definitions. See also *page set*.

C

CCSID. Coded character set identifier.

CDF. Channel definition file.

channel. See *message channel*.

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.

checkpoint. A time when significant information is written on the log. Contrast with *syncpoint*. In MQSeries on UNIX systems, the point in time when a data record described in the log is the same as the data record in the queue. Checkpoints are generated automatically and are used during the system restart process.

CI. Control interval.

circular logging. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, the process of keeping all restart data in a ring of log files. Logging fills the first file in the ring and then moves on to the next, until all the files are full. At this point, logging goes back to the first file in the ring and starts again, if the space has been freed or is no longer needed. Circular logging is used during restart recovery, using the log to roll back transactions that were in progress when the system stopped. Contrast with *linear logging*.

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.

cluster. A network of queue managers that are logically associated in some way.

coded character set identifier (CCSID). The name of a coded set of characters and their code point assignments.

command. In MQSeries, an administration instruction that can be carried out by the queue manager.

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*.

Common Run-Time Environment (CRE). A set of services that enable system and application programmers to write mixed-language programs. These shared, run-time services can be used by C, COBOL85, FORTRAN, Pascal, and TAL programs.

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.

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 command. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, a command that can be entered interactively from the operating system command line. Such a command requires only that the MQSeries product be installed; it does not require a special utility or program to run it.

control interval (CI). A fixed-length area of direct access storage in which VSAM stores records and creates distributed free spaces. The control interval is the unit of information that VSAM transmits to or from direct access storage.

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.

CRE. Common Run-Time Environment.

Cross Systems Coupling Facility (XCF). Provides the OS/390 coupling services that allow authorized programs in a multisystem environment to communicate with programs on the same or different OS/390 systems.

D

data bag. In the MQAI, a bag that allows you to handle properties (or parameters) of objects.

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.

DCE. Distributed Computing Environment.

DCI. Data conversion interface.

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.

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.

dual logging. A method of recording MQSeries for OS/390 activity, where each change is recorded on two data sets, so that if a restart is necessary and one data set is unreadable, the other can be used. Contrast with *single logging*.

dual mode. See *dual logging*.

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*.

ESM. External security manager.

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 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.

external security manager (ESM). A security product that is invoked by the OS/390 System Authorization Facility. RACF is an example of an ESM.

F

FIFO. First-in-first-out.

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)

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

Generalized Trace Facility (GTF). An OS/390 service program that records significant system events, such as supervisor calls and start I/O operations, for the purpose of problem determination.

get. In message queuing, to use the MQGET call to remove a message from a queue. See also *browse*.

GTF. Generalized Trace Facility.

I

ICE. Intersystem Communications Environment is a family of Tandem-based software products that enables you to access a variety of applications on Tandem computers.

ILE. Integrated Language Environment®.

in-doubt unit of recovery. In MQSeries, the status of a unit of recovery for which a syncpoint has been requested but not yet confirmed.

Integrated Language Environment (ILE). The AS/400 Integrated Language Environment. This replaces the AS/400 Original Program Model (OPM).

initialization input data sets. Data sets used by MQSeries for OS/390 when it starts up.

initiation queue. A local queue on which the queue manager puts trigger messages.

installable services. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, additional functionality provided as independent components. The installation of each component is optional: in-house or third-party components can be used instead. See also *authorization service*, *name service*, and *user identifier service*.

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.

Interactive System Productivity Facility (ISPF). An IBM licensed program that serves as a full-screen editor and dialog manager. It is used for writing application programs, and provides a means of generating standard screen panels and interactive dialogues between the application programmer and terminal user.

Internet Protocol (IP). A protocol used to route data from its source to its destination in an Internet environment. This is the base layer, on which other protocol layers, such as TCP and UDP are built.

IP. Internet Protocol.

ISPF. Interactive System Productivity Facility.

L

linear logging. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, the process of keeping restart data in a sequence of files. New files are added to the sequence as necessary. The space in which the data is written is not reused until the queue manager is restarted. Contrast with *circular logging*.

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, to enable them to recover in the event of failure.

log control file. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, the file containing information needed to monitor the use of log files (for example, their size and location, and the name of the next available file).

log file. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, a file in which all significant changes to the data controlled by a queue manager are recorded. If the primary log files become full, MQSeries allocates secondary log files.

logical unit of work (LUW). See *unit of work*.

LU 6.2. A type of logical unit (LU) that supports general communication between programs in a distributed processing environment.

M

MCA. Message channel agent.

MCI. Message channel interface.

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 at one end and a receiver at the other end) 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. See also *message queue interface*.

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.

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.

MQAI. MQSeries Administration Interface.

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 Administration Interface (MQAI). A programming interface to MQSeries.

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 object that contains a list of names, for example, queue names.

name service. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, the facility that determines which queue manager owns a specified queue.

name service interface (NSI). The MQSeries interface to which customer- or vendor-written programs that resolve queue-name ownership must conform. A part of the MQSeries Framework.

NetBIOS. Network Basic Input/Output System. An operating system interface for application programs used on IBM personal computers that are attached to the IBM Token-Ring Network.

nonpersistent message. A message that does not survive a restart of the queue manager. Contrast with *persistent message*.

NSI. Name service interface.

O

OAM. Object authority manager.

object. In MQSeries, an object is a queue manager, a queue, a process definition, a channel, a namelist, or a storage class (OS/390 only).

object authority manager (OAM). In MQSeries on UNIX systems, MQSeries for AS/400, and MQSeries for Windows NT, the default authorization service for command and object management. The OAM can be replaced by, or run in combination with, a customer-supplied security service.

Open Transaction Manager Access (OTMA). A transaction-based, connectionless client/server protocol. It functions as an interface for host-based communications servers accessing IMS TM applications through the OS/390 Cross Systems Coupling Facility (XCF). OTMA is implemented in an OS/390 sysplex environment. Therefore, the domain of OTMA is restricted to the domain of XCF.

OPM. Original Program Model

Original Program Model (OPM). The AS/400 Original Program Model. This is no longer supported on MQSeries. It is replaced by the Integrated Language Environment (ILE).

OTMA. Open Transaction Manager Access.

output log-buffer. In MQSeries for OS/390, a buffer that holds recovery log records before they are written to the archive log.

P

page set. A VSAM data set used when MQSeries for OS/390 moves data (for example, queues and messages) from buffers in main storage to permanent backing storage (DASD).

PCF. Programmable command format.

PCF command. See *programmable command format*.

performance event. A category of event indicating that a limit condition has occurred.

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.

point of recovery. In MQSeries for OS/390, the term used to describe a set of backup copies of MQSeries for OS/390 page sets and the corresponding log data sets required to recover these page sets. These backup copies provide a potential restart point in the event of page set loss (for example, page set I/O error).

principal. In MQSeries on UNIX systems, MQSeries for OS/2 Warp, and MQSeries for Windows NT, a term used for a user identifier. Used by the object authority manager for checking authorizations to system resources.

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.

queuing. See *message queuing*.

R

RBA. Relative byte address.

reason code. A return code that describes the reason for the failure or partial success of an MQI call.

recovery log. In MQSeries for OS/390, data sets containing information needed to recover messages, queues, and the MQSeries subsystem. MQSeries for OS/390 writes each record to a data set called the *active log*. When the active log is full, its contents are off-loaded to a DASD or tape data set called the *archive log*. Synonymous with *log*.

relative byte address (RBA). The displacement in bytes of a stored record or control interval from the beginning of the storage space allocated to the data set to which it belongs.

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.

request message. A type of message used to request a reply from another program.

RESLEVEL. In MQSeries for OS/390, an option that controls the number of CICS user IDs checked for API-resource security in MQSeries for OS/390.

resource. Any facility of the computing system or operating system required by a job or task. In MQSeries for OS/390, examples of resources are buffer pools, page sets, log data sets, queues, and messages.

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.

Resource Recovery Services (RRS). An OS/390 facility that provides 2-phase syncpoint support across participating resource managers.

return codes. The collective name for completion codes and reason codes.

rollback. Synonym for *back out*.

RRS. Resource Recovery Services.

S

SAF. System Authorization Facility.

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.

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*.

signaling. In MQSeries for OS/390 and MQSeries for Windows 2.1, a feature that allows the operating system to notify a program when an expected message arrives on a queue.

single logging. A method of recording MQSeries for OS/390 activity where each change is recorded on one data set only. Contrast with *dual logging*.

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.

SPX. Sequenced Packet Exchange transmission protocol.

storage class. In MQSeries for OS/390, a storage class defines the page set that is to hold the messages for a particular queue. The storage class is specified when the queue is defined.

subsystem. In OS/390, a group of modules that provides function that is dependent on OS/390. For example, MQSeries for OS/390 is an OS/390 subsystem.

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 Authorization Facility (SAF). An OS/390 facility through which MQSeries for OS/390 communicates with an external security manager such as RACF.

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

TACL. Tandem Advanced Command Language.

TCP. Transmission Control Protocol.

TCP/IP. Transmission Control Protocol/Internet Protocol.

thread. In MQSeries, the lowest level of parallel execution available on an operating system platform.

time-independent messaging. See *asynchronous messaging*.

TMF. Transaction Management Facility.

TMI. Trigger monitor interface.

TM/MP. NonStop Transaction Manager/MP.

trace. In MQSeries, a facility for recording MQSeries activity. The destinations for trace entries can include GTF and the system management facility (SMF). See also *global trace* and *performance trace*.

Transmission Control Protocol (TCP). Part of the TCP/IP protocol suite. A host-to-host protocol between hosts in packet-switched communications networks. TCP provides connection-oriented data stream delivery. Delivery is reliable and orderly.

Transmission Control Protocol/Internet Protocol (TCP/IP). A suite of communication protocols that support peer-to-peer connectivity functions for both local and wide area networks.

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

UDP. User Datagram Protocol.

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*.

User Datagram Protocol (UDP). Part of the TCP/IP protocol suite. A packet-level protocol built directly on the Internet Protocol layer. UDP is a connectionless and less reliable alternative to TCP. It is used for application-to-application programs between TCP/IP host systems.

user identifier service (UIS). In MQSeries for OS/2 Warp, the facility that allows MQI applications to associate a user ID, other than the default user ID, with MQSeries messages.

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

XCF. Cross Systems Coupling Facility.

Bibliography

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

MQSeries cross-platform publications

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

- MQSeries for AIX V5.1
- MQSeries for AS/400 V5.1
- MQSeries for AT&T GIS UNIX V2.2
- MQSeries for Compaq (DIGITAL) OpenVMS V2.2.1.1
- MQSeries for DIGITAL UNIX (Compaq Tru64 UNIX) V2.2.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for OS/390 V2.1
- MQSeries for SINIX and DC/OSx V2.2
- MQSeries for Sun Solaris V5.1
- MQSeries for Tandem NonStop Kernel V2.2.0.1
- MQSeries for VSE/ESA V2.1
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1
- MQSeries for Windows NT V5.1

Any exceptions to this general rule are indicated.

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

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 Queue Manager Clusters

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

This book applies to the following MQSeries products only:

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

MQSeries Clients

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

MQSeries System Administration

The *MQSeries System Administration* book, SC33-1873, supports day-to-day management of local and remote MQSeries objects. It includes topics such as security, recovery and restart, transactional support, problem

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

This book applies to the following MQSeries products only:
- MQSeries for AIX V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1

MQSeries Command Reference

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

MQSeries Programmable System Management

The *MQSeries Programmable System Management* book, SC33-1482, provides both reference and guidance information for users of MQSeries events, Programmable Command Format (PCF) messages, and installable services.

MQSeries Administration Interface Programming Guide and Reference

The *MQSeries Administration Interface Programming Guide and Reference*, SC34-5390, provides information for users of the MQAI. The MQAI is a programming interface that simplifies the way in which applications manipulate Programmable Command Format (PCF) messages and their associated data structures.

This book applies to the following MQSeries products only:
- MQSeries for AIX V5.1
- MQSeries for AS/400 V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1

MQSeries Messages

The *MQSeries Messages* book, GC33-1876, which describes "AMQ" messages issued by MQSeries, applies to these MQSeries products only:
- MQSeries for AIX V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1
- MQSeries for Windows V2.0
- MQSeries for Windows V2.1

This book is available in softcopy only.

For other MQSeries platforms, the messages are supplied with the system. They do not appear in softcopy manual form.

MQSeries Application Programming Guide

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

MQSeries Application Programming Reference

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

MQSeries Application Programming Reference Summary

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

MQSeries Using C++

MQSeries Using C++, SC33-1877, provides both guidance and reference information for users of the MQSeries C++ programming-language binding to the MQI. MQSeries C++ is supported by these MQSeries products:
- MQSeries for AIX V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for AS/400 V5.1
- MQSeries for OS/390 V2.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1

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

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

MQSeries Using Java

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

- MQSeries for AIX V5.1
- MQSeries for AS/400 V5.1
- MQSeries for HP-UX V5.1
- MQSeries for MVS/ESA V1.2
- MQSeries for OS/2 Warp V5.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1

This book is available in softcopy only.

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 V5.1 Quick Beginnings, GC33-1867

MQSeries for AS/400

MQSeries for AS/400 V5.1 Quick Beginnings, GC34-5557

MQSeries for AS/400 V5.1 System Administration, SC34-5558

MQSeries for AS/400 V5.1 Application Programming Reference (ILE RPG), SC34-5559

MQSeries for AT&T GIS UNIX

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

MQSeries for Compaq (DIGITAL) OpenVMS

MQSeries for Digital OpenVMS System Management Guide, GC33-1791

MQSeries for Digital UNIX (Compaq Tru64 UNIX)

MQSeries for Digital UNIX System Management Guide, GC34-5483

MQSeries for HP-UX

MQSeries for HP-UX V5.1 Quick Beginnings, GC33-1869

MQSeries for OS/2 Warp

MQSeries for OS/2 Warp V5.1 Quick Beginnings, GC33-1868

MQSeries for OS/390

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

MQSeries for OS/390 Version 2 Release 1 Program Directory

MQSeries for OS/390 System Management Guide, SC34-5374

MQSeries for OS/390 Messages and Codes, GC34-5375

MQSeries for OS/390 Problem Determination Guide, GC34-5376

MQSeries link for R/3

MQSeries link for R/3 Version 1.2 User's Guide, GC33-1934

MQSeries for SINIX and DC/OSx

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

MQSeries for Sun Solaris

MQSeries for Sun Solaris V5.1 Quick Beginnings, GC33-1870

MQSeries for Tandem NonStop Kernel

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

MQSeries for VSE/ESA

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

MQSeries for VSE/ESA System Management Guide, GC34-5364

MQSeries for Windows

MQSeries for Windows V2.0 User's Guide, GC33-1822

MQSeries for Windows V2.1 User's Guide, GC33-1965

MQSeries for Windows NT

MQSeries for Windows NT V5.1 Quick Beginnings, GC34-5389

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

MQSeries LotusScript Extension, SC34-5404

Softcopy books

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

BookManager® format

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

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

HTML format

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

- MQSeries for AIX V5.1
- MQSeries for AS/400 V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1 (compiled HTML)
- MQSeries link for R/3 V1.2

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

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

Portable Document Format (PDF)

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

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

`http://www.adobe.com/`

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

- MQSeries for AIX V5.1
- MQSeries for AS/400 V5.1
- MQSeries for HP-UX V5.1
- MQSeries for OS/2 Warp V5.1
- MQSeries for Sun Solaris V5.1
- MQSeries for Windows NT V5.1
- MQSeries link for R/3 V1.2

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

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

PostScript format

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

Windows Help format

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

MQSeries information available on the Internet

The MQSeries product family Web site is at:

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

By following links from this Web site you can:

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

Index

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To 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, to this address:

  Information Development Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–870229
  - From within the U.K., use 01962–870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC33-1349-08

Spine information:

MQSeries®

MQSeries Planning Guide