

**Chapter
1**

**CONCEPTS AND
FACILITIES**

*Get on the
Fast Track!*



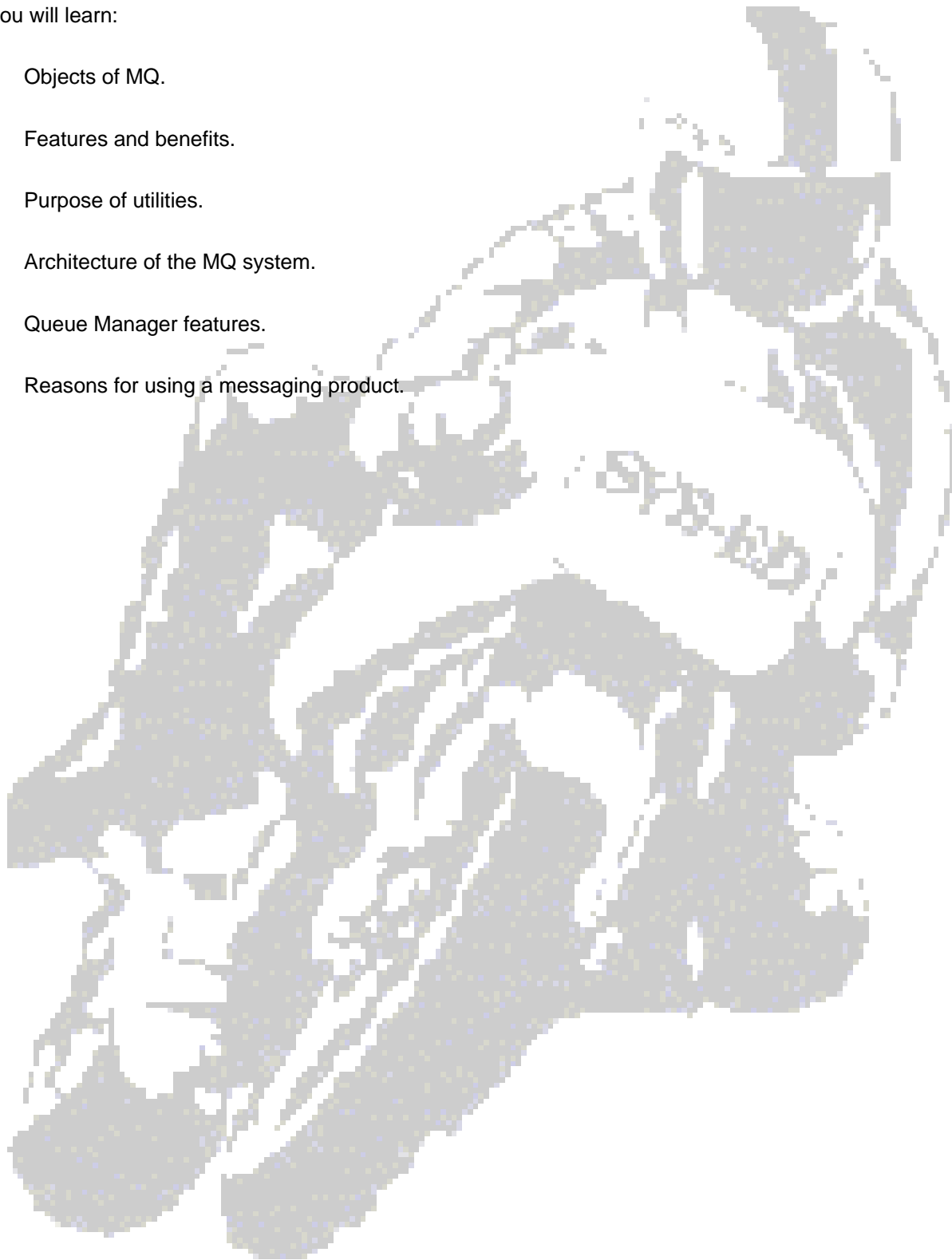
TM

**SYS-ED/
Computer
Education
Techniques, Inc.**

Objectives

You will learn:

- Objects of MQ.
- Features and benefits.
- Purpose of utilities.
- Architecture of the MQ system.
- Queue Manager features.
- Reasons for using a messaging product.



1 Message

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

Messages are used to transfer information from:

- one application program to another.
- or
- to different parts of the same application.

The applications can be running on the same platform, or on different platforms.

WebSphere MQ messages have two parts:

Application data	The content and structure of the application data which is defined by the application programs that use them.
Message descriptor	Identifies the message and contains additional control information, such as the type of message and the priority assigned to the message by the sending application.

2 Queue

A queue is a data structure used to store messages until they are retrieved by an application. Queues are managed by a queue manager.

The queue manager is responsible for maintaining the queues it owns, storing all the messages it receives onto the appropriate queues, and retrieving the messages in response to application requests.

The messages can be placed on the queues by application programs, or by a queue manager as part of its operation.

3 WebSphere MQ Object

Many of the tasks carried out when using WebSphere MQ involve manipulating WebSphere MQ objects.

In WebSphere MQ for z/OS the object types are:

- queues
- processes
- authentication information objects
- namelists
- storage classes
- coupling facility structures
- channels
- clusters
- system objects
- queue manager security objects

The manipulation or administration of objects includes:

- Starting and stopping queue managers.
- Creating objects, particularly queues, for applications.
- Working with channels to create communication paths to queue managers on other (remote) systems.

4 Queue Manager

Before application programs can use WebSphere MQ on a z/OS system, it will be necessary to install the WebSphere MQ for z/OS product and start a queue manager.

The queue manager owns and manages the set of resources that are used by WebSphere MQ including:

- Page sets that hold the WebSphere MQ object definitions and message data.
- Logs that are used to recover messages and objects in the event of queue manager failure.
- Processor storage.
- Connections through which different application environments (CICS, IMS, and Batch) can access the WebSphere MQ API.
- The WebSphere MQ channel initiator, which allows communication between WebSphere MQ on a z/OS system and other systems.

The queue manager has a name, and applications can connect to it using this name.

5 Queue Manager Subsystem

On z/OS, WebSphere MQ runs as a z/OS subsystem that is started at IPL time.

Within the subsystem, the queue manager is started by executing a JCL procedure that specifies the z/OS data sets that contain information about the logs, and that hold object definitions and message data (the page sets).

The subsystem and the queue manager have the same name, of up to four characters.

All queue managers in the network must have unique names, even if they are on different systems, sysplexes, or platforms.

6 Shared Queues

Queues can be non-shared, owned by and accessible to only one queue manager, or shared, owned by a queue-sharing group.

A queue-sharing group consists of a number of queue managers, running within a single z/OS sysplex, that can access the same WebSphere MQ object definitions and message data concurrently. Within a queue-sharing group, the shareable object definitions are stored in a shared DB2 database, and the messages are held inside one or more Coupling Facility structures.

The shared DB2 database and the Coupling Facility structures are resources that are owned by several queue managers.

7 Page Sets and Buffer Pools

When a message is put on to a non-shared queue, the queue manager stores the data on a page set in such a way that it can be retrieved when a subsequent operation gets a message from the same queue.

If the message is removed from the queue, space in the page set that holds the data is subsequently freed for reuse.

As the number of messages held on a queue increases, so the amount of space in the page set holding message data increases, and as the number of messages on a queue reduces, the space used in the page set reduces.

To reduce the overhead of writing data to and reading data from the page sets, the queue manager buffers the updates into processor storage.

The amount of storage used to buffer the page set access is controlled through WebSphere MQ objects called buffer pools.

8 Logging

Any changes to objects held on page sets, and operations on persistent messages, are recorded as log records. These log records are written to a log data set called the active log.

The name and size of the active log data set is held in a data set called the bootstrap data set (BSDS).

When the active log data set fills up, the queue manager switches to another log data set so that logging can continue, and copies the content of the full active log data set to an archive log data set. Information about these actions, including the name of the archive log data set, is held in the bootstrap data set.

9 Tailoring the Queue Manager Environment

When the queue manager is started, a set of initialization parameters that control how the queue manager operates are read. In addition, data sets containing WebSphere MQ commands are read, and the commands they contain are executed.

Typically, these data sets contain definitions of the system objects required for WebSphere MQ to run, and you can tailor these to define or initialize the WebSphere MQ objects necessary for your operating environment.

When these data sets have been read, any objects defined by them are stored, either on a page set or in DB2.

10 Recovery and Restart

At any time during the operation of WebSphere MQ, there might be changes held in processor storage that have not yet been written to the page set. These changes are written out to the page set that is the least recently used by a background task within the queue manager.

If the queue manager terminates abnormally, the recovery phase of queue manager restart can recover the lost page set changes because persistent message data is held in log records. This means that WebSphere MQ can recover persistent message data and object changes right up to the point of failure.

If a queue manager that is a member of a queue-sharing group encounters a Coupling Facility failure, the persistent messages on that queue can only be recovered if the Coupling Facility structure has been backed up.

11 Channel Initiator

The channel initiator provides and manages resources that enable WebSphere MQ distributed queuing. WebSphere MQ uses Message Channel Agents (MCAs) to send messages from one queue manager to another.

To send messages from queue manager A to queue manager B, a sending MCA on queue manager A must set up a communications link to queue manager B. A receiving MCA must be started on queue manager B to receive messages from the communications link. This one-way path consisting of the sending MCA, the communications link, and the receiving MCA is known as a channel.

- The sending MCA takes messages from a transmission queue and sends them down a channel to the receiving MCA.
- The receiving MCA receives the messages and puts them on to the destination queues.

In WebSphere MQ for z/OS, the sending and receiving MCAs all run inside the channel initiator. The channel initiator runs as a z/OS address space under the control of the queue manager.

There can only be a single channel initiator connected to a queue manager and it is run inside the same z/OS image as the queue manager. There can be thousands of MCA processes running inside the channel initiator concurrently.

The channel initiator also contains other processes concerned with the management of the channels.

Listeners	Listens for inbound channel requests on a communications subsystem such as TCP, and start a named MCA when an inbound request is received.
Supervisor	Manages the channel initiator address space, it is responsible for restarting channels after a failure.
Name server	Used to resolve TCP names into addresses.
SSL tasks	Used to perform encryption and decryption and check certificate revocation lists.

12 WebSphere MQ Utilities

WebSphere MQ for z/OS provides a set of utility programs in order to perform various administrative tasks.

These utilities include:

- Performing backup, restoration, and reorganization tasks.
- Issuing commands and process object definitions.
- Generating data-conversion exits.
- Modifying the bootstrap data set.
- Listing information about the logs.
- Printing the logs.
- Setting up DB2 tables and other DB2 utilities.
- Processing messages on the dead-letter queue.

12.1 CSQUTIL Utility

The CSQUTIL utility program is provided with WebSphere MQ for z/OS in order to help in performing backup, restoration, and reorganization tasks, and issue commands and process object definitions.

This utility program can be used for invoking the following functions:

Function	Description
COMMAND	To issue MQSC commands, to record object definitions, and to make client-channel definition files.
COPY	To read the contents of a named WebSphere MQ for z/OS 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 sets.
EMPTY	To delete the contents of a named WebSphere MQ for z/OS message queue or the contents of all the queues of a named page set, retaining the definitions of the queues.
FORMAT	To format WebSphere MQ for z/OS page sets.
LOAD	To restore the contents of a named WebSphere MQ for z/OS message queue or the contents of all the queues of a named page set from a sequential file created by the COPY function.
PAGEINFO	To extract page set information from one or more page sets.
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.

12.1 Change Log Inventory Utility

The WebSphere MQ for z/OS 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.

12.2 Print Log Map Utility

CSQJU004 is the WebSphere MQ for z/OS print log map utility program. It runs as a stand-alone utility for listing:

- 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.

12.3 Log Print Utility

CSQ1LOGP is the log print utility program; it is run as a stand-alone utility.

The utility can be run specifying:

- A bootstrap data set – BSDS.
- Active logs - with no BSDS.
- Archive logs with no BSDS.

12.4 Queue-sharing Group Utility

CSQ5PQSG queue-sharing group utility program runs as a stand-alone utility to set up DB2 tables and perform other DB2 tasks required for queue-sharing groups.

12.5 Active Log Preformat Utility

CSQJUFMT active log preformat utility formats active log data sets before they are used by a queue manager.

If the active log data sets are preformatted by the utility, log write performance is improved on the queue manager's first pass through the active logs.

12.6 Dead-Letter Queue Handler Utility

CSQUDLQH dead-letter queue handler utility program runs as a stand-alone utility. It checks messages that are on the dead-letter queue and processes them according to a set of rules that are supplied to the utility.

13 Message Queuing

The MQSeries products enable programs to communicate with one another across a network of dissimilar components using a consistent application programming interface.

Applications using this interface are known as message queuing applications, as they use the messaging and queuing style:

Messaging	Programs communicate by sending each other data in messages rather than calling each other directly.
Queuing	Messages are placed on queues in storage, allowing programs to run independently of each other, at different speeds and times, in different locations, and without having a logical connection between them.

Message queuing is used in electronic mail. In a queuing system, messages are stored at intermediate nodes until the system is ready to forward them. At their final destination they are stored in an electronic mailbox until the addressee is ready to read them. If one part of the system suffers a problem, many parts of the system become unusable.

In a message queuing environment, each program from the set that makes up an application suite is designed to perform a well-defined, self-contained function in response to a specific request. To communicate with another program, a program must put a message on a predefined queue. The other program retrieves the message from the queue, and processes the requests and information contained in the message.

Queuing is the mechanism by which messages are held until an application is ready to process them.

Queuing provides the capability for:

- Communicating between programs without having to write the communication code.
- Selecting the order in which a program processes messages.
- Balancing loads on a system by arranging for more than one program to service a queue.
- Increasing the availability of applications by arranging for an alternative system to service the queues when the primary system is unavailable.

14 Message in Message Queuing

In message queuing, a message is a collection of data sent by one program and intended for another program.

MQSeries defines four types of messages:

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

14.1 Message Descriptor

An MQSeries message consists of control information and application data. The control information is defined in a message descriptor structure (MQMD) and contains:

- The type of the message.
- An identifier for the message.
- The priority for delivery of the message.

The structure and content of the application data is determined by the participating programs, not by MQSeries.

A message channel agent moves messages from one queue manager to another.

14.2 Message Queue

A message queue is a named destination to which messages can be sent. Messages accumulate on queues until they are retrieved by programs that service those queues. Queues reside in, and are managed by, a queue manager.

The physical nature of a queue depends on the operating system on which the queue manager is running. A queue can either be a volatile buffer area in the memory of a computer or a data set on a permanent storage device such as a disk.

The physical management of queues is the responsibility of the queue manager and is not made apparent to the participating application programs.

15 Queue Manager

A queue manager is a system program that provides queuing services to applications. It provides an application programming interface so that programs can put messages on and get messages from queues.

A queue manager provides additional functions for:

- creating new queues.
- altering the properties of existing queues.
- controlling the operation of the queue manager.

To an application, each queue manager is identified by a connection handle. For a program to use the services of a queue manager, it must establish a connection to that queue manager.

For applications to be able to send messages to applications that are connected to other queue managers, the queue managers must be able to communicate among themselves. MQSeries implements a store-and-forward protocol to ensure the delivery of messages between applications.

16 Cluster

A cluster is a network of queue managers that are logically associated in some way.

When a queue manager needs to send messages to another it must have defined a transmission queue, a channel to the remote queue manager, and a remote queue definition for every queue to which it wants to send messages.

If queue managers are grouped into a cluster, the queue managers can make the queues that they host available to every other queue manager in the cluster. Then, assuming that the necessary network infrastructure is 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.

There are two different reasons for using clusters:

- Reducing system administration.
- Improving availability and workload balancing.

As soon as a cluster is established, there will be a benefit from simplified system administration. Queue managers that are part of a cluster need fewer definitions; accordingly, there will be a reduced risk of making an error in definitions.

17 Shared Queue, Queue-Sharing Group, and Intra-group Queuing

Shared queues, queue-sharing groups, and intra-group queuing are only available on MQSeries for OS/390, V2.2 with sysplex.

A shared queue is a type of local queue whose messages can be accessed by one or more queue managers that are in a sysplex.

The queue managers that can access the same set of shared queues form a queue-sharing group (QSG). They communicate with each other by means of a coupling facility (CF) that stores the shared queues.

Message transfer between queue managers in a queue-sharing group is intra-group queuing (IGQ). It provides the capability for performing fast message transfer without defining channels.

An MQSeries client is an independently installable component of an MQSeries product. It allows MQSeries applications, to be run by means of a communications protocol to interact with one or more Message Queue Interface (MQI) servers on other platforms and connect to their queue managers.

18 Message Queuing: Main Features

The main features of applications that use message queuing techniques are:

- There are no direct connections between programs.
- Communication between programs can be time-independent.
- Work can be performed by small, self-contained programs.
- Communication can be driven by events.
- Applications can assign a priority to a message.
- Security.
- Data integrity.
- Recovery support.

No direct connections between programs

Message queuing is a technique for indirect program-to-program communication. It can be used within any application where programs communicate with each other.

There are no physical connections between programs that communicate using message queues. A program sends messages to a queue owned by a queue manager, and another program retrieves messages from the queue

Time-independent communication

Programs requesting others to do work do not have to wait for the reply to a request.

Small programs

Message queuing provides the capability for using small, self-contained programs. Instead of a single, large program performing all the parts of a job sequentially, a job can be spread out over several smaller, independent programs.

Event-driven processing

Programs which can be controlled according to the state of queues.

Message priority

A program can assign a priority to a message when it puts the message on a queue. This determines the position in the queue at which the new message is added.

Security

Authorization checks are carried out on each resource, using the tables that are set up and maintained by the MQSeries administrator. RACF or other external security managers may be used within MQSeries for OS/390. Within MQSeries on UNIX systems, AS/400, Compaq (DIGITAL) OpenVMS, Tandem NonStop Kernel, and Windows NT, a security manager, the Object Authority Manager (OAM), is provided as an installable service. By default, the OAM is active.

On VSE/ESA, security is provided by CICS.

Data integrity

Data integrity is provided via units of work. Syncpoint support operates either internally or externally to MQSeries depending on the form of syncpoint coordination selected for the application.

Recovery support

For recovery to be possible, all persistent MQSeries updates will need to be logged. In the event that recovery is necessary, all persistent messages will be restored., all in-flight transactions will be rolled back and any syncpoint commit and backouts will be handled with the syncpoint manager in control.

18.1 Message Queuing: Benefits

The benefits of message queuing are:

- Applications can be designed using small programs that can be shared between many applications.
- New applications can be built by reusing the building blocks.
- Applications written to use message queuing techniques are not affected by changes in the way queue managers work.
- Communication protocols do not need to be used.
- Programs that receive messages do not need to be running at the time messages are sent to them; the messages are retained on queues.
- The cost of designing applications can be reduced; development is faster.

19 MQSeries for OS/390, z/OS

With MQSeries for OS/390, applications can be written that:

- Use message queuing within CICS or IMS.
- Send messages between batch, CICS, and IMS applications.
- Send messages to applications that run on other MQSeries platforms.
- Process several messages together as a single unit of work that can be committed or backed out.
- Send messages to and interact with IMS applications by means of the IMS bridge.
- Participate in units of work coordinated by RRS.

Each environment within OS/390 has its own characteristics, advantages, and disadvantages. The advantage of MQSeries for OS/390 is that applications are not tied to any one environment, but can be distributed to take advantage of the benefits of each environment.

Examples:

End-user interfaces can be developed using TSO or CICS.

Processing-intensive modules can be run in OS/390 batch.

Database applications can be run in IMS or CICS.

20 MQSeries for non-OS/390, z/OS Platforms

With MQSeries for non-OS/390 platforms, applications can be written that:

- Send messages to other applications running under the same operating systems.
- Send messages to applications that run on other MQSeries platforms.
- Use message queuing from within CICS Transaction Server for OS/2, CICS for AS/400, TXSeries for AIX, TXSeries for HP-UX, CICS for Siemens Nixdorf, SINIX, TXSeries for Sun Solaris, and TXSeries for Windows NT, DOS, and Windows 3.1 applications.
- Use message queuing from within Encina for AIX, HP-UX, SINIX, Sun Solaris, and Windows NT.
- Use message queuing from within Sybase for AIX, Sun Solaris, and Windows NT.

21 Designing Messages

A message is created when using an MQI call to put the message on a queue. As input to the call, control information is specified in a message descriptor (MQMD) and the data that is to be sent to another program.

Questions to Consider

The type of message that should be used.
--

Is this a simple application in which a message can be sent, and then no further action is taken.

Is the application asking for a reply to a question.
--

If a question is being asked then the name of the queue which is receiving the reply should be included in the message descriptor.
--

Are the request and reply messages to be synchronous?

This implies that a timeout period for the reply to answer a request has been set.
--

If a reply is not received within that period, it is treated as an error.

Is the preference to work asynchronously?

In this case processes do not have to depend upon the occurrence of specific events, such as common timing signals?

Should different priorities be assigned to some of the messages that have been created.

A priority value can be assigned to each message, and the queue defined so that it maintains its messages in order of their priority. If this is done, when another program retrieves a message from the queue, it always gets the message with the highest priority.

If the queue does not maintain its messages in priority order, a program that retrieves messages from the queue will retrieve them in the order in which they were added to the queue.
--

Will messages be discarded when the queue manager restarts?

The queue manager preserves all persistent messages, recovering them when necessary from the MQSeries log files, when it is restarted.

Nonpersistent messages and temporary dynamic queues are not preserved. Any messages that are not to be discarded must be defined as persistent at the time they are created.

When writing an application for MQSeries for OS/2 Warp, MQSeries for Windows NT, or MQSeries on UNIX systems, it is important to be knowledgeable with how a log file allocation is set up in order to reduce the risk of designing an application that will run to the log file limits.

Messages on shared queues, which are only available on MQSeries for OS/390, V2.2 systems with sysplex) cannot be made persistent; therefore, they will not be preserved when the queue manager restarts. However, messages on shared queues are held in the Coupling Facility (CF), and are lost only if the CF fails.

Do I want to give information about myself to the recipient of my messages?

Normally, the queue manager sets the user ID, but authorized applications can also set this field, thereby providing the capability for including user ID and other information that the receiving program can use for accounting or security purposes.

21.1 Waiting for Messages

A program that is serving a queue can await messages by:

- Making periodic calls on the queue in order to ascertain whether a message has arrived (polling).
- Waiting until either a message arrives, or a specified time interval expires.
- Setting a signal so that the program is informed when a message arrives.

21.2 Correlating Replies

In MQSeries applications, when a program receives a message that asks it to do some work, the program usually sends one or more reply messages to the requester. In order to help the requester to associate these replies with its original request, an application can set a correlation identifier field in the descriptor of each message.

21.3 Starting MQSeries Programs Automatically

MQSeries triggering enables a program to be started automatically when messages arrive on a queue.

Trigger conditions can be set on a queue in order that a program is started to process that queue:

- Every time a message arrives on the queue.
- When the first message arrives on the queue.
- When the number of messages on the queue reaches a predefined number.

22 Generating MQSeries Reports

The following reports can be requested within an application:

- Exception reports
- Expiry reports
- COA: Confirm-on-arrival reports
- COD: Confirm-on-delivery reports
- PAN: Positive action notification reports
- NAN: Negative action notification reports

22.1 Application Programming

MQSeries supports the IBM Message Queue Interface (MQI) and the Application Messaging Interface (AMI).

- The MQI includes a set of calls with which can send and receive messages, and manipulate MQSeries objects.
- The Application Messaging Interface (AMI), a simpler interface than the MQI, may be sufficient in some cases.

23 Call Interface

The MQI calls provide the capability for:

- Connecting programs to, and disconnecting programs from, a queue manager.
- Opening and closing objects such as queues, queue managers, namelists, and processes.
- Putting messages on queues.
- Receiving messages from a queue, or browsing them and leaving them on the queue.
- Inquiring about the attributes/properties of MQSeries objects, and setting some of the attributes of queues.
- Committing and backing out changes made within a unit of work, in environments where there is no natural syncpoint support..
- Coordinating queue manager updates and updates made by other resource managers

The MQI provides structures, which are groups of fields, with which to supply input to, and get output from, the calls. It also provides a large set of named constants to help in supplying options in the parameters of the calls.