

**Chapter
2**

MESSAGES

*Get on the
Fast Track!*



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

Objectives

You will learn:

- MQSeries messages.
- Message types.
- Report message types.
- Format of message control information and message data.
- Format of message data.
- Application data conversion.
- Message priorities.
- Message persistence.
- Messages that fail to be delivered.
- Messages that are backed out.
- Reply-to queue and Queue Manager
- Message context.

1 MQSeries Messages

An MQSeries message consists of two parts:

- Message descriptor
- Application data

The application data carried in an MQSeries message is not changed by a queue manager; unless data conversion is carried out on it. MQSeries does not put any restrictions on the content of this data.

The length of the data in each message cannot exceed the value of the MaxMsgLength attribute of both the queue and queue manager.

In MQSeries for AIX, MQSeries for AS/400, MQSeries for HP-UX, MQSeries for OS/2 Warp, MQSeries for Sun Solaris, MQSeries for Compaq Tru64 UNIX, and MQSeries for Windows NT, the MaxMsgLength defaults to 100 MB.

In MQSeries for OS/390 Version 2.2, the MaxMsgLength attribute of the queue manager is fixed at 100 MB and the MaxMsgLength attribute of the queue defaults to 4 MB (4 194 304 bytes) which can be adjusted up to a size of 100 MB

However, the limit is 63 KB when using shared queues accessed by queue managers in a queue-sharing group; which is only available on MQSeries for OS/390, V2.2 with sysplex.

2 Types of Messages

There are four types of messages defined by MQSeries:

- Datagram
- Request
- Reply
- Report

Applications can use the first three types of messages to pass information between themselves. The fourth type, report, is for applications and queue managers to use for reporting information about events such as the occurrence of an error.

Each type of message is identified by an MQMT_* value. It is also possible to define a message types.

2.1 Datagrams

A datagram should be used when a reply is not required from the application that receives the message; the message is gotten from the queue.

Such an application is unlikely to request an acknowledgement for a message because it probably does not matter if a message is not delivered.

The application will send an update message after a short period of time.

2.2 Request Messages

A request message should be used in order to reply from the application that receives the message.

In order to link a reply message with a request message, there are two options:

- Provide the application with the responsibility of ensuring that it puts information into the reply message that relates to the request message.

- The report field can be used in the message descriptor of the request message in order to specify the content of the Msgld and Correlld fields of the reply message:
 - A request can be made in order that either the Msgld or the Correlld of the original message is to be copied into the Correlld field of the reply message. The default action is to copy Msgld.
 - A request can be made in order that either a new Msgld is generated for the reply message, or that the Msgld of the original message is to be copied into the Msgld field of the reply message. The default action is to generate a new message identifier).

2.3 Reply Messages

A reply message should be used for replying to another message. When creating a reply message, it is recommended that any options that were set in the message descriptor of the message to which the reply is being made be respected.

Report options specify the content of the message identifier and correlation identifier fields.

These fields allow the application that receives the reply to correlate the reply with its original request.

2.4 Report Messages

Report messages inform applications about events such as the occurrence of an error when processing a message.

They can be generated by:

- A queue manager
 - A message channel agent
- Or
- An application

Report messages can be generated at any time, and they may arrive on a queue when an application is not expecting them.

3 Report Message Types

When putting a message on a queue, a selection can be made to receive:

Message	Explanation
An exception report message	This is sent in response to a message that had the exceptions flag set. It is generated by the message channel agent (MCA) or the application.
An expiry report message	This indicates that an application attempted to retrieve a message that had reached its expiry threshold; the message is marked to be discarded. This type of report is generated by the queue manager.
COA: confirmation of arrival report message	This indicates that the message has reached its target queue. It is generated by the queue manager. This indicates that the message has been retrieved by a receiving application. It is generated by the queue manager.
PAN: positive action notification report message	This indicates that a request has been successfully serviced; that the action which has been requested in the message has been performed successfully. This type of report is generated by the application.
NAN: negative action notification report message	This indicates that a request has not been successfully serviced; that the action requested in the message has not been performed successfully. This type of report is generated by the application.

Each type of report message contains one of the following:

- The entire original message.
- The first 100 bytes of data in the original message.
- No data from the original message.

3.1 Format of Message Control Information and Message Data

The queue manager is only interested in the format of the control information within a message, whereas applications that handle the message are interested in the format of both the control information and the data.

4 Format of Message Control Information

Control information in the character-string fields of the message descriptor must be in the character set used by the queue manager. The CodedCharSetId attribute of the queue manager object defines this character set.

Control information must be in this character set because when applications pass messages from one queue manager to another, message channel agents that transmit the messages use the value of this attribute to determine what data conversion they must perform.

5 Format of Message Data

The format of message data can be specified in the following formats:

- The format of the application data.
- The character set of the character data.
- The format of numeric data.

5.1 Format

The format indicates to the receiver of a message the format of the application data in the message.

When the queue manager creates a message, in some circumstances it uses the Format field to identify the format of that message. When a queue manager cannot deliver a message, it puts the message on a dead-letter queue, which is for undelivered messages. It then adds a header which contains more control information to the message, and changes the Format field to show this.

The queue manager has a number of built-in formats with names beginning with "MQ". Additional formats can be defined; but names beginning with "MQ" should not be used.

When creating and using formats, data-conversion exits must be written to support a program getting the message using MQGMO_CONVERT.

5.2 CodedCharSetId

The CodedCharSetId defines the character set of character data in the message. If this character is to be set to that of the queue manager, this field can be set to the constant MQCCSI_Q_MGR or MQCCSI_INHERIT.

When getting a message from a queue, the value of the CodedCharSetId field should be compared with the value that the application is expecting. If the two values differ, it may be necessary to convert any character data in the message or use a data-conversion message exit if one is available.

5.3 Encoding

Encoding describes the format of numeric message data that contains binary integers, packed-decimal integers, and floating point numbers. It is usually encoded according to the particular machine on which the queue manager is running.

When putting a message on a queue, normally the constant MQENC_NATIVE should be specified in the Encoding field. This means that the encoding of the message data will be the same as that of the machine on which the application is running.

6 Application Data Conversion

Application data may need to be converted to the character set and the encoding which is required by another application where different platforms are concerned. It can be converted at the sending queue manager, or at the receiving queue manager.

If the library of built-in formats does not meet specific requirements; it is possible to define create definitions. The type of conversion depends on the message format which is specified in the format field of the message descriptor, MQMD.

6.1 Conversion at the Sending Queue Manager

The CONVERT channel attribute must be set to YES if it will be necessary to send message channel agent, MCA, to convert the application data.

The conversion is performed at the sending queue manager for certain built-in formats and for user-defined formats if a suitable user exit is supplied.

Messages with MQFMT_NONE specified are not converted.

6.2 Conversion at the Receiving Queue Manager

Application message data may be converted by the receiving queue manager for the built-in formats and user-defined formats. The conversion is performed during the processing of an MQGET call if the MQGMO_CONVERT option is specified.

7 Message Priorities

The priority of a message, in the Priority field of the MQMD structure, is set when the message is put on a queue.

A numeric value can be set for the priority, or the message can be allowed to take the default priority of the queue.

The MsgDeliverySequence attribute of the queue determines whether messages on the queue are stored in FIFO or in FIFO within priority sequence.

If this attribute is set to MQMDS_PRIORITY, messages are enqueued with the priority specified in the Priority field of their message descriptors; but if it is set to MQMDS_FIFO, messages are enqueued with the default priority of the queue.

Messages of equal priority are stored on the queue in order of arrival. The DefPriority attribute of a queue sets the default priority value for messages being put on that queue. This value is set when the queue is created, but it can be changed afterwards. Alias queues, and local definitions of remote queues, may have different default priorities from the base queues to which they resolve.

The value of the MaxPriority attribute of the queue manager is the maximum priority that can be assigned to a message processed by that queue manager. The value of this attribute can not be changed. In MQSeries, the attribute has the value 9; messages can be created having priorities between 0, which is the lowest, and 9, which is the highest.

8 Message Persistence

Persistent messages are written out to logs and queue data files. If a queue manager is restarted after a failure, it recovers these persistent messages as necessary from the logged data.

Messages that are not persistent are discarded if a queue manager stops, whether the stoppage is as a result of an operator command or because of the failure of some part of the system.

When creating a message, if the message descriptor, MQMD, has been initialized using the defaults, the persistence for the message will be taken from the DefPersistence attribute of the queue specified in the MQOPEN command.

Alternatively, the persistence of the message can be set using the Persistence field of the MQMD structure to define the message as persistent or not persistent. The performance of an application is affected when using persistent messages; the extent of the effect depends on the performance characteristics of the machine's I/O subsystem and how the syncpoint options are used on each platform:

A persistent message, outside the current unit of work, is written to disk on every put and get operation.

In MQSeries on UNIX systems, MQSeries for Compaq (DIGITAL) OpenVMS, MQSeries for OS/390, MQSeries for OS/2 Warp, MQSeries for VSE/ESA, and MQSeries for Windows NT, a persistent message within the current unit of work is logged only when the unit of work is committed. The unit of work could contain many queue operations.

Nonpersistent messages can be used for fast messaging if retrieved outside syncpoint.

9 Messages that Fail to be Delivered

When a queue manager is unable to put a message on a queue, there will be various options.

The following options can be implemented:

- Attempt to put the message on the queue again.
- Request that the message is returned to the sender.
- Put the message on the dead-letter queue.

10 Messages that are Backed Out

When processing messages from a queue under the control of a unit of work, the unit of work could consist of one or more messages.

- If a backout occurs, the messages which were retrieved from the queue are reinstated on the queue, and they can be processed again in another unit of work.
- If the processing of a particular message is causing the problem, the unit of work is backed out again. This could cause a processing loop.

Messages which were put to a queue are removed from the queue.

11 Reply-To Queue and Queue Manager

There are occasions when messages may be received in response to a message that has been sent:

- A reply message in response to a request message.
- A report message about an unexpected event or expiry.
- A report message about a COA -Confirmation Of Arrival or a COD - Confirmation Of Delivery event.
- A report message about a PAN -Positive Action Notification or a NAN - Negative Action Notification event.

The MQMD structure can be used for specifying the name of the queue to which reply and report messages are sent, in the ReplyToQ field. Specify the name of the queue manager that owns the reply-to queue in the ReplyToQMgr field.

12 Message Context

Message context information allows the application that retrieves the message to find out about the originator of the message.

The retrieving application may want to:

- Check that the sending application has the correct level of authority.
- Perform some accounting function in order that it can charge the sending application for any work it has to perform
- Keep an audit trail of all the messages it has worked with.

When using the MQPUT or MQPUT1 call to put a message on a queue, the queue manager can be specified to add some default context information to the message descriptor. Applications that have the appropriate level of authority can add extra context information.

All context information is stored in the eight context fields of the message descriptor. The type of information falls into two categories:

- Identity.
- origin context information.

12.1 Identity Context

Identity context information identifies the user of the application that first put the message on a queue:

- The queue manager fills the UserIdentifier field with a name that identifies the user -the way that the queue manager can do this depends on the environment in which the application is running.
- The queue manager fills the AccountingToken field with a token or number that it determined from the application that put the message.
- Applications can use the ApplIdentityData field for any extra information that they want to include about the user (for example, an encrypted password).

12.2 Origin Context

Origin context information describes the application that put the message on the queue on which the message is currently stored.

The message descriptor contains the following fields for origin context information:

Field	Description
PutApplType	The type of application that put the message.
PutApplName	The name of the application that put the message.
PutDate	The date on which the message was put on the queue.
PutTime	The time at which the message was put on the queue.
ApplOriginData	Any extra information that an application may want to include.