Rocket Model 204

MQ/204 Reference

Version 7 Release 5.0

September 2014
204–75–MQ-01

Notices

Edition

Publication date: September 2014
Book number: 204–75–MQ-01
Product version: Version 7 Release 5.0

Copyright

Trademarks

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Corporate Information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA

Contacting Technical Support

If you have current support and maintenance agreements with Rocket Software and CCA, contact Rocket Software Technical support by email or by telephone:

Email: m204support@rocketsoftware.com

Telephone :

North America	+1.800.755.4222
United Kingdom/Europe	+44 (0) 20 8867 6153

Alternatively, you can access the Rocket Customer Portal and report a problem, download an update, or read answers to FAQs. You will be prompted to log in with the credentials supplied as part of your product maintenance agreement.

To log in to the Rocket Customer Portal, go to:

www.rocketsoftware.com/support

Contents

About this Manual

1 Working with MQ/204

Index

About this Manual

This manual describes MQ/204, which permits a Model 204 Online or batch job running under either z/OS to access the IBM z/OS WebSphere MQ software.

Audience

This manual is intended for WebSphere MQ and Model 204 application developers, who want to deliver and accept Model 204 data using the Message Queuing Middleware functionality of WebSphere MQ for program-to-program communication. Familiarity with WebSphere MQ and Model 204 terminology and functionality is assumed.

A note about User Language and SOUL

Model 204 version 7.5 provides a significantly enhanced, object-oriented, version of User Language called SOUL. All existing User Language programs will continue to work under SOUL, so User Language can be considered to be a subset of SOUL, though the name "User Language" is now deprecated. In this manual, the name "User Language" has been replaced with "SOUL."

Model 204 documentation set

To access the Rocket Model 204 documentation, see the Rocket Documentation Library (http://docs.rocketsoftware.com/), or go directly to the Rocket Model 204 documentation wiki (http://m204wiki.rocketsoftware.com/).

Additional documentation

Depending on your level of experience and familiarity with WebSphere MQ and Model 204, you might need additional documentation. For WebSphere MQ documentation, contact your IBM representative.

Rocket recommends the following IBM manuals:

- *WebSphere MQ Application Programming Guide* (SC34-6064)
- *WebSphere MQ Application Programming Reference* (SC34-6062)
- *WebSphere MQ for z/OS Messages and Codes* V5.3.1 (SC34-6056)

To obtain additional information on WebSphere MQ or to download the WebSphere MQ manuals, access the IBM Web site. Their Web address is:

`http://www-4.ibm.com/software/ts/mqseries/`

Documentation conventions

This manual uses the following standard notation conventions in statement syntax and examples:

Convention	Description
`TABLE`	Uppercase represents a keyword that you must enter exactly as shown.
`TABLE` *`tablename`*	In text, italics are used for variables and for emphasis. In examples, italics denote a variable value that you must supply. In this example, you must supply a value for *tablename*.
`READ [SCREEN]`	Square brackets ([ ]) enclose an optional argument or portion of an argument. In this case, specify READ or READ SCREEN.
`UNIQUE | PRIMARY KEY`	A vertical bar (|) separates alternative options. In this example, specify either UNIQUE or PRIMARY KEY.
`TRUST | NOTRUST`	Underlining indicates the default. In this example, NOTRUST is the default.
`IS {NOT | LIKE}`	Braces ({ }) indicate that one of the enclosed alternatives is required. In this example, you must specify either IS NOT or IS LIKE.
`item ...`	An ellipsis (. . .) indicates that you can repeat the preceding item.
`item ,...`	An ellipsis preceded by a comma indicates that a comma is required to separate repeated items.
`All other symbols`	In syntax, all other symbols (such as parentheses) are literal syntactic elements and must appear as shown.
`nested-key` `::=` *`column_name`*	A double colon followed by an equal sign indicates an equivalence. In this case, *nested-key* is equivalent to *column_name*.
`Enter your account:` `sales11`	In examples that include both system-supplied and user-entered text, or system prompts and user commands, boldface indicates what you enter. In this example, the system prompts for an account and the user enters **sales11**.
File > Save As	A right angle bracket (>) identifies the sequence of actions that you perform to select a command from a pull-down menu. In this example, select the Save As command from the File menu.
E`DIT`	Partial bolding indicates a usable abbreviation, such as E for EDIT in this example.

1

Working with MQ/204

MQ/204 is the Model 204 interface to a WebSphere MQ-enabled application. SOUL (User Language) extensions allow you to send and receive messages and process them using SOUL operators and data structures, so that Synchronous program-to-program communications may be established between Model 204 and any other WebSphere MQ-enabled application running on the same or different CPU and using the same or different platform.

MQ/204 allows you to define and reference all WebSphere MQ objects using internal names and spares SOUL programmers from knowing details of WebSphere MQ communications. Simple and effective MQ/204 command and operator syntax resembles the WebSphere MQ mnemonics but free you from internal details, so that minimal WebSphere MQ knowledge is required to program MQ/204-based application. When MQ/204 defines a message queue or a queue manager, it is actually referring to a previously defined WebSphere MQ message queue or queue manager.

Using MQ/204, you can separate application programs so that the program sending a message can continue processing without waiting for a reply from the receiver. Messages are forwarded by WebSphere MQ agents, so that sending and receiving programs are completely independent from each other and may or may not be active at the same time.

WebSphere MQ architecture

The IBM WebSphere MQ product manages queues of messages. Like an e-mail system, WebSphere MQ lets an application put and get messages on WebSphere MQ queues; the sender and recipient do not have to be active at the same time.

Figure 1-1 shows WebSphere MQ providing one or more queue managers that are system processes. Each queue manager controls one or more queues. To

access a queue, connect to the appropriate queue manager, then open the desired queue.

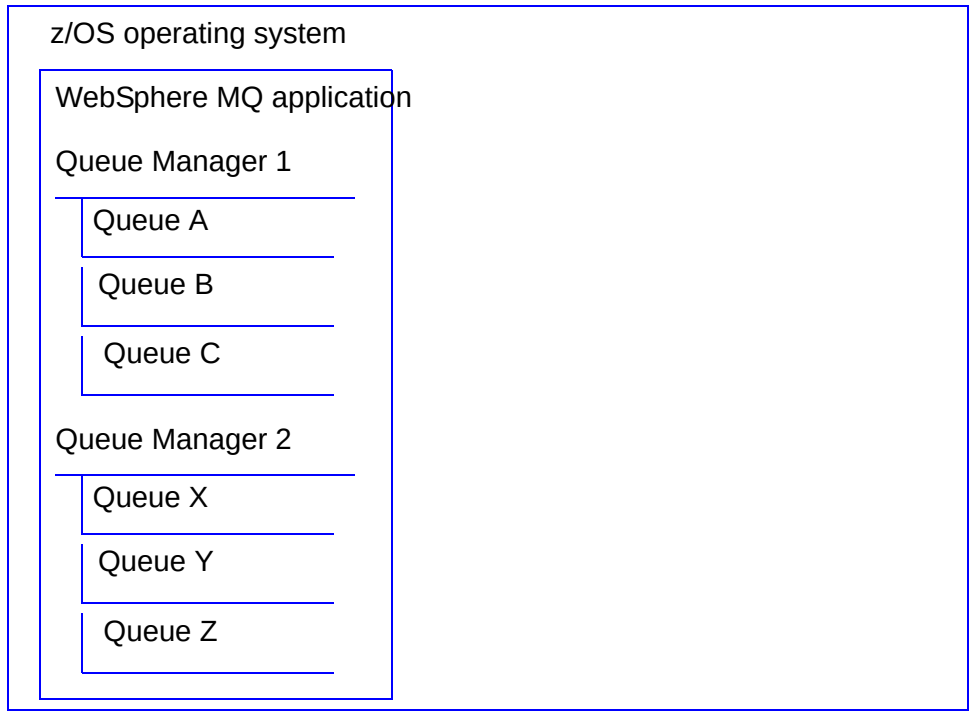

Figure 1-1. WebSphere MQ architecture

Types of WebSphere MQ implementation for z/OS

Although the WebSphere MQ for z/OS also comes with a TSO and CICS option, MQ/204 implements the z/OS batch option of the WebSphere MQ, which has the following implications:

- The z/OS batch option supports multiple queue managers.
- The z/OS batch option does not support two-phase commits.

z/OS performance considerations

System programmers for z/OS may observe delayed release of CSA storage by IBM WebSphere MQ.

MQ/204 environment requirements

The MQ/204 environment requires that you run the following minimum versions of software:

- Model 204 Version 6 Release 1.0
- WebSphere MQ Version 5.x

- MQMD Version 2

MQ/204 architecture

The z/OS WebSphere MQ provides one or more queue managers that are system processes. Each queue manager controls one or more queues. To access a queue, connect to the appropriate queue manager, then open the desired queue.

Operating-system subtasks, as shown in Figure 1-2, issue WebSphere MQ API calls for MQ/204.

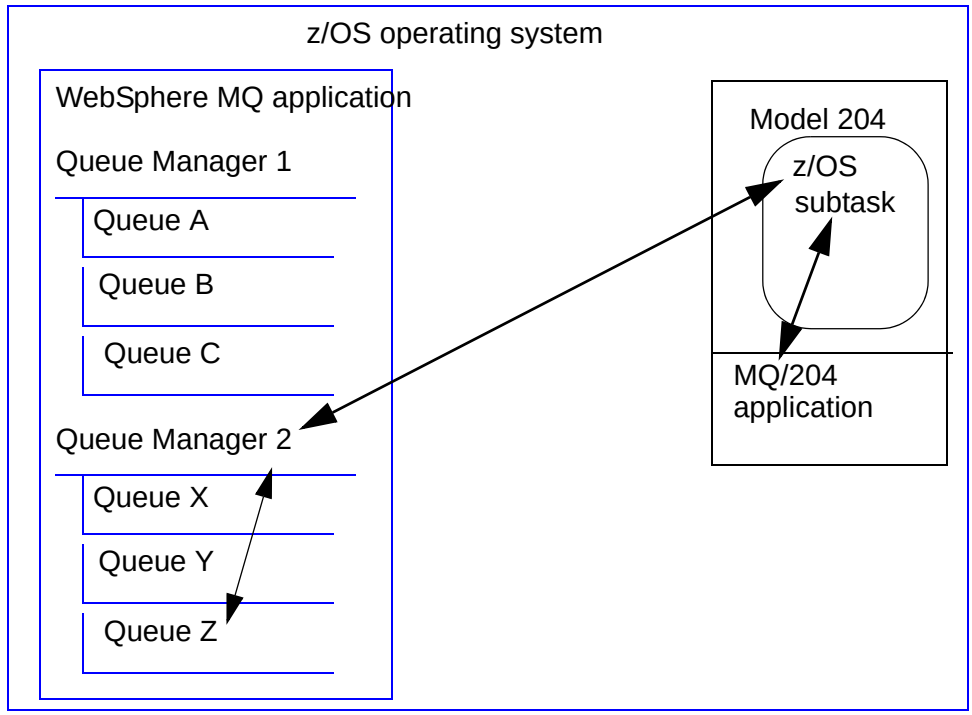

Figure 1-2. Model 204 makes gets and puts to a WebSphere MQ application via z/OS system subtasks

MQ/204 requires an z/OS subtask, because all calls to WebSphere MQ are synchronous. Other Model 204 users can continue to work a set of z/OS subtasks to perform all needed communications with WebSphere MQ, thus isolating the Model 204 task to process other users.

Subtask management

WebSphere MQ API calls make use of a pool of operating-system subtasks to communicate. The size of this pool is governed by the parameters MQINTASK (initial size of subtask pool) and MQMXTASK (maximum size of subtask pool). During system initialization, a pool of MQINTASK subtasks is allocated. Additional subtasks are allocated dynamically as needed during Online execution, up to a maximum of MQMXTASK subtasks.

Each subtask can be in one of the following states:

State	Description
Free and unconnected	Not being used, and not connected to a queue manager.
In use	In use by a user thread.
Free and connected	Available and connected to a queue manager. This state permits keeping connections to queue managers active over multiple uses of a subtask as a performance optimization.

Subtask allocation

When an application needs a connection to a new queue manager, MQ/204 utilizes the algorithm in Figure 1-3 to determine how to assign a subtask,

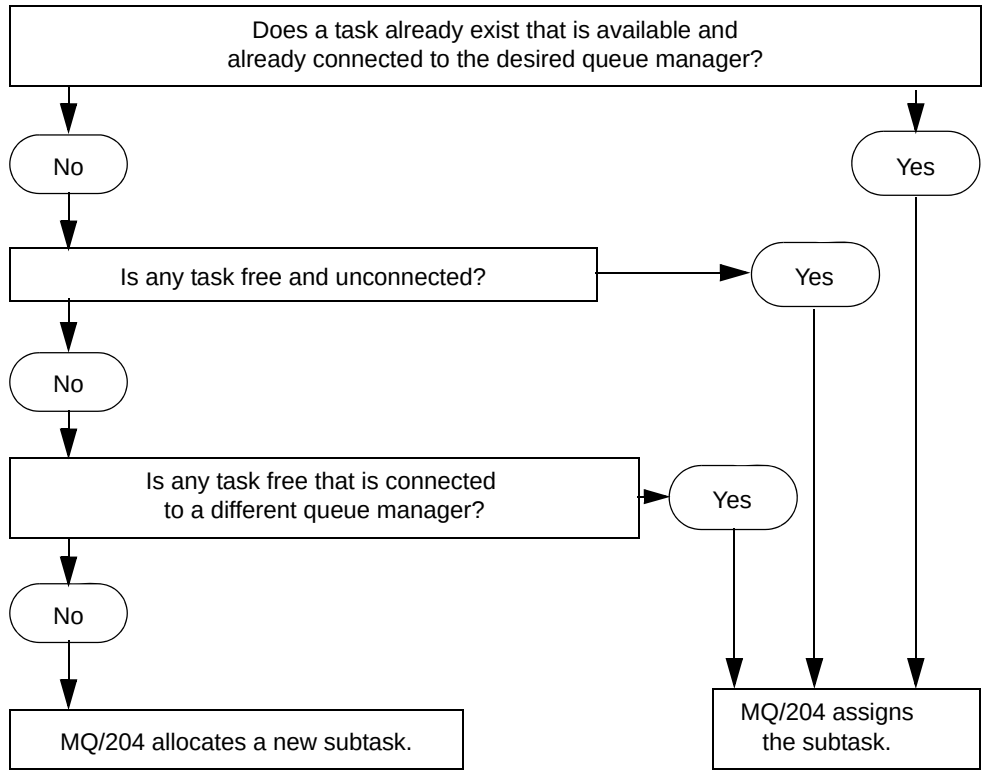

Figure 1-3. Algorithm used to assign a subtask

MQ/204 continues to allocate new subtasks according to this algorithm until MQMXTASK subtasks are assigned. Once the maximum number of subtasks is allocated, applications wait for up to MQWAIT milliseconds for an existing subtask to become available. Either a subtask frees up before the wait time expires, or a “no-subtasks-available” error is returned to the SOUL program in the $STATUS return code.

Subtask freeing

When freeing a subtask, MQ/204 tries to keep its queue manager connection, removing it from the in-use pool and adding it to the free-and-connected pool. MQ/204 disconnects a subtask from the queue manager and adds it to the free-and-unconnected pool only if the queue manager is stopped.

If the user is bumped and the MQ operation has not finished, the MQ subtask performing the operation is placed in the “delayed detach” state. The count of available subtasks is decremented and all associations with the user are removed. When the operation has finished, the subtask is detached and the MQDELDTP PST (pseudo subtask) runs to free the related storage areas.

MQ/204 queue management

MQ/204 enables a Model 204 Online or batch job running under z/OS to access the IBM z/OS WebSphere MQ communication facilities. All the queues and queue managers available to MQ/204 are originally defined by the WebSphere MQ software.

Determining message destination

When a WebSphere MQ application, such as MQ/204, sends a message, WebSphere MQ must know the final destination of the message. The final destination is identified by the combination of a queue manager name, called the target queue manager, and a queue name, as one of the following categories. WebSphere MQ determines which category a queue is in at the time the queue is opened.

Local queues

An MQ/204 application delivers a message to a queue by connecting directly to the target queue manager, that is, the queue manager that owns the target queue. The queue is described as a local queue from the perspective of the application sending the message. The MQ/204 application needs to know the target queue manager name and the target queue name.

Remote queues

An MQ/204 application might send a message to a queue, but cannot connect directly to the target queue manager; for example, the target queue manager is running under a different OS on a different machine. In this case, the queue is described as a remote queue from the perspective of the application sending the message.

To deliver the message, the MQ/204 application connects directly to a local queue manager, called the source queue manager, and requests that the source queue manager pass the message to the target queue manager. The MQ/204 application must know the source queue manager name, the target

queue manager name, and the target queue name. In MQ/204, the source queue manager name is identified by the MQ/204 MQDEQMAN parameter.

Locally defined remote queues

As an alternative to remote queues, a queue manager can have queue definitions that point to remote queues, that is, queues owned by another queue manager. To deliver the message, the MQ/204 application connects directly to the local queue manager using the local name. From the point of view of the MQ/204 application, the queue behaves as if it were local. The MQ/204 application needs only the source queue manager name and the source queue name; the source queue manager knows the target queue manager name and the target queue name.

Cluster queues

Cluster queues are used in a similar way to locally defined remote queues, but require reduced WebSphere MQ administration. A cluster queue is a queue that is hosted by a cluster queue manager and made available to other queue managers in the cluster. The cluster queue manager makes a local queue definition for the queue, specifying the name of the cluster where the queue is to be found.

Determining message handling

WebSphere MQ allows messages to be sent via an MQPUT statement to local, remote, or locally defined remote queues, but allows messages to be received only via MQGET from local queues.

Rules for queue names

MQ/204 determines whether a queue is opened as local or remote based on the following rules for queue names:

- A queue name without an embedded colon (:) is an MQ/204 entity name that refers to either a local queue or a local definition of a remote queue.
- A queue name with an embedded colon (:) is interpreted as *QueueManagerName:QueueName:*
  - If no default queue manager is specified by the MQDEQMAN parameter, the open fails.
  - Otherwise, the queue is opened as a remote queue using the value in MQDEQMAN parameter as the local queue manager.

Model 204 support

MQ/204 has the following Model 204 support features:

- User 0 parameters within Model 204 allocate resources that support:

  - Connections to queue managers
  - Number of concurrent threads that can use this support

- Model 204 commands define queues and queue managers to Model 204 and control access to and monitoring of queue managers.

Using MQ/204, you can refer to queues and queue managers as entities, that is, by names that differ from their operating system level (external) names. Also, the MONITOR command permits the system manager to track the use of MQ/204. Commands and objects similar in syntax and function to the DEFINE commands used for Horizon identify queue managers and queues to Model 204.

Within SOUL, the following additional features are implemented:

- You can open and close WebSphere MQ queues with extensions to the OPEN and CLOSE statements, which is consistent with the Model 204 approach to interactions with external communication entities.
- You can reset defaults for open queues with an extension to the MODIFY statement.
- You can easily map between WebSphere MQ and SOUL, because the statements that manipulate queues are named like the WebSphere MQ API. For example:

SOUL statement	WebSphere MQ action
MQBACK	Backs out WebSphere MQ transactions
MQCMIT	Commits WebSphere MQ transactions
MQGET	Gets messages from a queue
MQPUT	Puts messages on an open queue
MQPUT1	Puts one message on a not-yet-opened queue

Remote queue support

You can put a message on remote queues that do not have corresponding DEFINE QUEUE commands in the Online. Remote queues might be owned by any queue manager on the network and are typically not on the same mainframe. Support for remote queues means that you can write MQ/204 applications that reply to any queues on the network, without issuing additional DEFINE QUEUE commands in Model 204.

To put a message on a remote queue with WebSphere MQ, you must open a remote queue using remote queue manager and remote queue names. MQ/204 connects you to the default queue manager, which should have all connections necessary to send your message to the remote queue. The next MQPUT statement you make sends your message to a remote queue.

Default queue manager

To support remote queues, a default queue manager is employed, because multiple queue managers can be defined to the Online. The default queue manager is the one to which you put messages when the destination queue is unknown to Model 204. The z/OS queue manager forwards the message to the queue manager that contains the destination remote queue. If the destination remote queue is unknown to the WebSphere MQ network, error information is returned to the SOUL program in $STATUS and $STATUSD return codes.

Local dynamic queue support

Sometimes you want an application to create a queue on an as-needed basis, which is called a local, dynamic queue. For example, if after a query the application has data to send, it creates a local dynamic queue; if there is no data to send, it does not create a queue. Also, when the queue is no longer needed it is closed and deleted.

So that an MQ/204 application can create a local dynamic queue, the WebSphere MQ system administrator defines and makes available a template known as a model queue. MQ/204 can create a local dynamic queue that takes the attributes of a WebSphere MQ model queue. You can request many attributes to define a model queue for your application. Of special interest is whether the preallocated model queue has specified DEFTYPE as permanent or temporary.

Local dynamic queues, either permanent or temporary, are created in your application by issuing an MQOPEN call with the name of a model queue.

Model queues can have one of the following DefinitionType attributes defined for local dynamic use

- MQQDT_PERMANENT_DYNAMIC
- MQQDT_TEMPORARY_DYNAMIC

This type of queue is deleted according to the rules that govern WebSphere MQ queues and queue managers. Please consult IBM WebSphere MQ documentation for details.

Reusing dynamic queue names

You can reuse the internal Model 204 name of a dynamic queue, after you issue a CLOSE QUEUE DELETE or CLOSE QUEUE DELETE_PURGE statement for that queue name. For example, to reuse the dynamic queue name, CCA1, issue the following statement:

```
CLOSE QUEUE CCA1 DELETE_PURGE
```

Rules of inheritance

The Model 204 commands, introduced in Chapter 3, and SOUL statements, introduced in Chapter 4, have many options that you can set. You can set options in the following ways:

- Set an option when you define a queue manager or queue. An option set at this point applies to all users. In the following definition, responses to messages are sent to QB, not back to the sending queue, QA:

  ```
  DEFINE QUEUE QA WITH SCOPE=SYSTEM QUEUEMANAGER=QM1 -
  REPLY_QUEUE=QB
  ```

- Change an option in individual MQ/204 statements on the fly. An option set at this point applies to just the issuing user. For the following MQPUT statement, the response to the message is sent to QC, not back to the default reply queue, QB.

  ```
  OPEN QA
  MQPUT MQ_BUFFER REPLY_QUEUE=QC
  ```

- Modify an option after you have opened a queue. An option set at this point applies to just the issuing user. If you frequently change the reply queue on MQPUT statements, you can separately modify the reply queue before issuing the MQPUT statements:

  ```
  MODIFY QUEUE QA REPLY_QUEUE=QC
  ```

Figure 1-4 illustrates the rules of inheritance, showing how the options are passed from the DEFINE QUEUE command to the OPEN and MQPUT1

statements. Options are then passed to the MQGET, MQPUT, and MODIFY QUEUE statements or you can change them in the same statements.

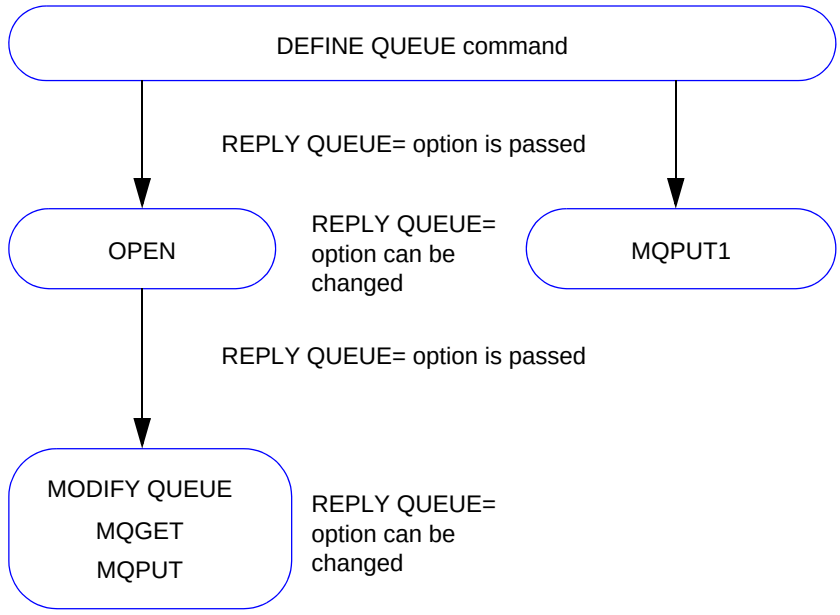

Figure 1-4. Rules of inheritance for passing options

The options correlate directly to options found in WebSphere MQ. See Table 5-1 on page 87 for a definition of each option and the command or statements that support it.

Reply queue and reply queue manager options

In MQ/204 you can indicate the reply queue and the reply queue manager in a DEFINE QUEUE command and OPEN QUEUE, MODIFY QUEUE, MQPUT, and MQPUT1 statements in one of the following ways:

- Specify a reply queue and reply queue manager explicitly using REPLY_QUEUE=*queue-name* and REPLY_QMGR=*queue-manager-name.*

  You can specify names as text strings, with or without quotation marks, of up to 48 characters or as a %variable containing text strings for SOUL statements. Use only external (real) names, as opposed to internal MQ/204 names that are specified in DEFINE QUEUE command and OPEN QUEUE statement.

- Specify a reply queue internal name only, without using the REPLY_QMGR option, to let MQ/204 find the real reply queue and reply queue manager names. Define the internal reply queue name by issuing the DEFINE QUEUE command for local queues, or the OPEN QUEUE statement for remote queues. In this case, MQ/204 finds the corresponding external queue and queue manager names and sets them in MQMD (message descriptor) structure. You can achieve the same result using the

REPLY_QMGR option with a zero-length name, for example, REPLY_QMGR = ''.

- Let WebSphere MQ resolve a queue manager name by setting the reply queue manager name to one or more blanks and providing the external queue name. If the reply queue name is a local definition of a remote queue:

Then WebSphere MQ sets the field…	To name of…	Otherwise, this field is…
MQMD.ReplyToQ	Remote queue	Not changed.
MQMD.ReplyToQMgr	Queue manager that owns the remote queue	Set to the name of the queue manager to which your application is connected.

Note: Never mix internal and external queue names in your applications—either by explicitly specifying them or by inheriting and implicitly mixing them—because, if the value of the REPLY_QMGR option is a name of nonzero length, MQ/204 does not substitute the external reply queue name for the internal name and sends the internal reply queue name instead of the external name.

Using runtime options

You can code MQ/204 statement options explicitly on a statement, in which case they are parsed at compile time. Alternatively, you can code them as runtime options, specified in syntax descriptions as ?%*variable,* which are parsed each time the statement is evaluated. You can specify both explicitly coded options and runtime options together in the same statement. The following MQ/204 statements support ?%variables:

- CLOSE QUEUE
- MODIFY QUEUE
- MQGET
- MQPUT
- MQPUT1
- OPEN QUEUE

Runtime options are specified in the syntax descriptions as ?%*variable*. The following “Rules that govern ?%variables” lists the features that apply to all ?%variables; see the individual statements in Chapter 4 for an explanation of syntax.

Rules that govern ?%variables

- Any option that is valid if specified at compilation can be specified in a run-time option variable.

- ?%Variable must be a string variable.
- At evaluation time, ?%variable:
  - Can be null.
  - Can contain multiple options.
  - Can contain options of the form *keyword=%variable,* however, the *%variable* must be a variable name in the program. It cannot be an image or a screen item.
  - Cannot contain keywords specified as %variables. For example, *%keyword=%variable* is invalid, just as for options specified at compile time.
- You can specify up to three runtime option variables on a statement, but each individual option specified at runtime must be wholly contained within one ?%variable and cannot span multiple ?%variables.
- Options specified within the ?%variables must be consistent. If they are inconsistent, a $STATUS code of 41 or 42 is returned.

  Examples of inconsistent use of options specified with a ?%variable are:
  - Using WAIT and NO_WAIT options together
  - Using the MODEL keyword without using the DYNAMICQNAME keyword.
- Options specified within the ?%variable override those specified on the statement at compile time.

Example The following example illustrates using two runtime options in an MQPUT statement:

```
%variable1 IS STRING LEN 255
%variable2 IS STRING LEN 255
%MSGTYPE IS STRING LEN 8

%variable1 = 'MSGTYPE=REPORT'
%variable2 = 'MSGTYPE=%MSGTYPE MSGID=DEF'

MQPUT 'MESSAGE' ON QUEUENAME MSGID=ABC ?%variable1
%MSGTYPE = 'REQUEST'
MQPUT 'MESSAGE' ON QUEUENAME ?%variable2
```

Parameters and task management

Parameters allocate z/OS system subtasks and other system resources for MQ/204.To use MQ/204, you must set MQINTASK to at least 1. However, more tasks might be needed. For each concurrent user who accesses queues, you need one task for each queue manager that the user accesses concurrently.

Parameters

While the Online is operating, you can view the MQ/204 parameters listed in Table 1-1.

Table 1-1. MQ/204 parameters

Parameter	Description	Default	Discussion	Setting can be changed…
MQDEQMAN	Identifies the WebSphere MQ default queue manager.	System default is not set	If set, its value is the name of the queue manager to use as the default queue manager. If not set, remote queue support is disabled; you must enter the actual name. Value must be the name of a previously defined queue manager; can be reset only to a previously defined queue manager.	By system manager following a DEFINE QM command
MQINTASK	Subtasks created during system initialization for managing connections to WebSphere MQ queue managers.	0	Set the value of MQINTASK to 1 (or more) to use MQ/204.	At initialization time, in CCAIN parameters
MQMXTASK	Maximum number of subtasks to use for WebSphere MQ connections.	Same as MQINTASK	You can set it to a greater value. If the value is greater than the value of MQINTASK, additional tasks are created as needed.	At initialization time, in CCAIN parameters
MQWAIT	Milliseconds that an OPEN QUEUE statement or a START QUEUEMANAGER command waits for the next available subtask.	0, meaning do not wait for tasks	The parameter applies if no more subtasks can be allocated because MQMXTASK subtasks are already allocated.	At initialization time, in CCAIN parameters; or by a system manager

Table 1-1. MQ/204 parameters (continued)

Parameter	Description	Default	Discussion	Setting can be changed…
UBUFSZ	Initial bytes per buffer that hold messages.	1024 bytes	Buffers contain the data portion of messages, not the MQMD message descriptor. Buffers are enlarged as needed. No data buffer is smaller than the value of UBUFSZ.	At initialization time, in CCAIN parameters

Task management

When a user finishes using a task, the task remains connected to a queue manager anticipating the next user, unless its queue manager is draining (see “STOP QUEUEMANAGER: Put a queue manager in drain state” on page 60). This technique optimizes performance for the common case where the next user of the task requires the same queue manager. Often a system has only one queue manager.

Accessing queues and queue managers

The WebSphere MQ administrator defines the queues and queue managers to WebSphere MQ. A Model 204 programmer with system administrator privileges can then issue the MQ/204 DEFINE QUEUEMANAGER and DEFINE QUEUE commands. This may be done in the startup deck for Model 204.

An application can reply only to request messages that specify a queue and queue manager defined by the system administrator to the Online, unless remote queue support is in use or local dynamic queues are being used.

After a connection to a queue manager is established, the queue manager enforces user ID based security when queues are opened. See “MODIFY QUEUE statement” on page 65.

MQ/204 security access control

The security of MQ/204 is as follows:

- SOUL programs can access a queue or queue manager after the system manager does one of the following:
  - Issues DEFINE commands
  - Enables remote queue support by setting the MQDEQMAN option
  - Defines the model queues required to support local dynamic queues.

- MQ/204 honors user-based security on queues that the WebSphere MQ system administrator set up outside Model 204.

WebSphere MQ API wait types access control

The WebSphere MQ API wait types, MQAPI and MQGWT, distinguish between waits for WebSphere MQ API execution and waits for a message to arrive on a queue:

- MQAPI waits encompass just the execution of the WebSphere MQ API.
- MQGWT waits include both WebSphere MQ API execution time and time spent waiting for a message to arrive on a queue.

Bumping users

A system manager can always displace a user who is waiting for a WebSphere MQ operation to complete or for a subtask to become available. MQ/204 waits are bumpable.

Triggering

Users can write their own trigger procedures by employing trigger queues and MQ/204 operators.

Queue security processing

When you open a queue, you undergo a user name authorization process. By default, with an z/OS batch WebSphere MQ connection, the job name is used. However, the MQOPEN facility permits you to pass an alternate user ID: the WebSphere MQ option MQOO_ALTERNATE_USER_AUTHORITY:

- If your site uses an external security package with Model 204, such as ACF2, Security Server (formerly RACF), or CA-Top Secret, then the Model 204 user ID is passed and the MQOO_ALTERNATE_USER_AUTHORITY option is set.
- If no security package is present, then the default mode of using the job name for security is used. The Model 204 user ID is not used unless it is authenticated by an external security package.
- If the PASS_USER_CONTEXT option is used with an OPEN QUEUE statement or an MQPUT1 statement, then the Model 204 user ID is passed with the MQOO_ALTERNATE_USER_AUTHORITY option set.

Data conversion

Specify message data format attributes using one of the following WebSphere MQ fields:

WebSphere MQ field	Indicates…
MQMD.Encoding	Storage formats used for: • Binary integers • Packed decimal integers • Floating point numbers
MQMD.CodedCharSetId	Character set used for string data

Data conversion formats

WebSphere MQ supports message data conversion through conversion exits that are invoked during MQGET processing. A conversion exit is invoked if the MQGMO_CONVERT option is specified on the MQGET call, and if the MQMD.Encoding field or the MQMD.CodedCharSetId field of the received message has a different value than was specified in the MQMD passed to MQGET. The name of the conversion exit to invoke is specified in the MQMD.Format field.

WebSphere MQ and MQ/204 can handle data conversion in the following ways.

Pass the data from the sender to the receiver

In this method, nothing is done to the data; it is simply passed from one application to the other. Data conversion is handled by the sending or receiving applications. The default value for WebSphere MQ MQMD.Format field is MQFMT_NONE (blanks), which tells WebSphere MQ not to convert data, even if there is a discrepancy in the character set or number encoding used.

Convert EBCDIC to ASCII

WebSphere MQ has built-in conversion for messages composed entirely of character string data. To send a message between z/OS and any ASCII machine and have character data automatically converted between EBCDIC and ASCII, specify the MQFMT_STRING constant in the WebSphere MQ MQMD.Format field of an MQPUT or MQPUT1 statement. This corresponds to specifying FORMAT='MQSTR' on a SOUL MQPUT or MQPUT1 statement. (MQFMT_STRING is a WebSphere MQ constant that equates to 'MQSTR'.)

Convert data by calling a conversion exit

If you want to convert messages that combine character string and numeric data in a single message, you must provide a data conversion exit for any

receiving machine that uses a different character set or number encoding. To call the conversion exit, set the FORMAT option in the MQPUT or MQPUT1 statement:

`FORMAT=`*`conversion_exit_name`*

For all MQ/204 MQGET and MQPUT calls:

Value for	Which provides…	Is specified in…
MQENC_NATIVE	Native number encoding	MQMD.Encoding field
MQCCSI_Q_MGR	Queue manager's coded character set identifier	MQMD.CodedCharSetId

On all MQGET calls, the MQGMO_CONVERT option is specified by default. This means that data conversion is controlled entirely through the FORMAT keyword of the MQPUT and MQPUT1 statements. An application program placing a message on the queue must specify:

- Message format.
- Name of the conversion exit required by the receiver of the message.
- If suppression of conversion is desired. If so, the NO_CONVERT option must be specified on the MQGET statement. See "Removing messages that do not convert" on page 36.

Data handling

Buffers for message data areas

Buffers are automatically allocated, initially at UBUFSZ value, and then enlarged as needed. The size of a user's buffer can be reset by issuing a MODIFY BUFFER statement.

MQGET, MQPUT, and MQPUT1 statements need a buffer to hold the data area of the message:

If message data is specified as…	Then the buffer must be at least the size of
%Variable or string constant	Variable or string constant
Image	Largest fixed (known at compile time) layout of the image: • Fixed layout of an image contains no dependences or unknown items. • If no layout of the image has a fixed size, then the buffer must be at least the size of the parameter UBUFSZ.

Enlarging the buffer

When the existing buffer is not large enough to hold the data area for MQPUT or MQPUT1 statements, MQ/204 makes it larger while preserving the existing buffer content. Once enlarged, the buffer remains the enlarged size until:

- MODIFY BUFFER statement with SIZE option is executed.
- MQPUT or MQPUT1 needs an even larger buffer.
- MQGET with BUFLEN indicates a larger buffer.

Controlling message context information

WebSphere MQ collects two kinds of message context information, each a set of fields in the message description. See “MQPUT statement” on page 75 for a greater understanding of the Descriptor field.

- Identity context identifies the following:

Descriptor field	Stores…
UserIdentifier	User who originally put the message on a queue
AccountingToken	Accounting token associated with the application and/or user that originally put the message on a queue
ApplIdentityData	Information that is defined by the application suite to provide additional information about the message or its originator. The queue manager treats this information as character data, but does not define the format of it. If the application does not define the value, the queue manager generates this information as a blank.

- Origin context identifies the following:

Descriptor field	Stores…
PutApplType	Type of application that put the message on the queue.
PutApplName	Name of the application that put the message on the queue.
PutDate	Date the message was put on the queue.
PutTime	Time the message was put on the queue.

Descriptor field	Stores…
ApplOrginData	Information that is defined by the application when it puts the message. For example, ApplOrginData could be set by suitably authorized applications to indicate whether the identity data is trusted. The queue manager treats this information as character data, but does not define the format of it. If the application does not specify a value, the queue manager generates this information as a blank.

For a discussion of message context, see the *WebSphere MQ Application Programming Guide*. See also the discussion of the following MQOPEN options in the *WebSphere MQ Application Programming Reference*:

```
MQOO_PASS_ALL_CONTEXT

MQOO_PASS_IDENTITY_CONTEXT

MQOO_SAVE_ALL_CONTEXT

MQOO_SET_ALL_CONTEXT

MQOO_SET_IDENTITY_CONTEXT
```

Inheriting and passing context information

You can use the DEFINE QUEUE command and the MODIFY QUEUE, MQPUT, MQPUT1, and OPEN QUEUE statements in application programs to manage message context information.

To pass context information, specify a queue that is open for input and has the SAVE_ALL_CONTEXT option.

The MQPUT, MQPUT1, and OPEN QUEUE statements have several options that you can set to pass context information. The context queue can either be specified on the statement that is passing context or be inherited. The inheritance rules for passing context are as follows:

- MQPUT1 and OPEN QUEUE statements inherit from DEFINE QUEUE command.
- MQPUT statement inherits from OPEN QUEUE statement.

If a context queue is not specified or inherited, then a statement attempting to pass context fails with a nonzero $STATUS value.

Dealing with messages larger than one image

An image is the SOUL equivalent of a data structure. When using a single image for message data, you must have an image that can contain the largest possible message you expect to retrieve or place. That is, space for the largest

possible message must be available in the user's server, because images are stored in either the full-screen buffer table, FSCB, or the global table, GTBL.

Image lists help manage the space required by using multiple images to hold a single message, so that you require a smaller Model 204 server.

Image lists

You can specify an image on MQGET, MQPUT, and MQPUT1 statements as a source or target for the data area of a message. You can also specify the source or target as a comma-separated list of up to ten images. For example:

```
MQGET image1, image2, image3 FROM queue_options

MQPUT image1, image2 ON queue_options

MQPUT1 image1, image2 ON queue_options
```

The images on the list are swapped in and processed serially. This provides server relief if the images are not global, because FSCB can be sized at the size of the largest image in the list instead of the largest message. The size of the MQ/204 message buffer is based on the sum of the lengths of all the images in the list.

On Get operations, the READLEN of each image is set. If the data runs out before filling all the images, the READLEN is set to zero for any images without data.

Alternatives to image lists

The MODIFY MQ_BUFFER, READ IMAGE, and WRITE IMAGE statements with the POSITION option provide alternatives for dealing with the server size issue; they can be used instead of image lists.

You can decide if you want to move messages out of the buffer programmatically, or have the MQ/204 statements handle messages in the buffer, so that the program never accesses the buffer. Directly manipulating the MQ_BUFFER area is programmatically more complex, but provides a finer level of access to message. For most applications, direct manipulation of the MQ_BUFFER area is not necessary.

WebSphere MQ transactions

Use MQCMIT and MQBACK statements to commit and back out WebSphere MQ transactions, called SYNCPOINT updates, to queues that belong to specific queue managers.

WebSphere MQ transactions are completely independent from Model 204 file update transactions. In addition, WebSphere MQ transactions against different queue managers are completely independent of one another.

Request cancellation and user restarts

If a cancellation or restart occurs, then all queues are closed, and any incomplete WebSphere MQ transactions are backed out.

End-of-request processing

The following table describes how end-of-request processing is handled:

If the request is…	Then…
Not part of a subsystem, or is part of an APSY AUTOCOMMIT=YES subsystem	All open queues are closed, and any uncommitted WebSphere MQ transactions against all queue managers that you are accessing are committed.
Part of an APSY AUTOCOMMIT=NO subsystem	WebSphere MQ transaction status is unchanged by end-of-request processing.

Leaving subsystems

When you leave a subsystem, end-of-request processing as described in the previous End-of-request processing section is done first. If WebSphere MQ transactions are active, which can happen if APSY AUTOCOMMIT=NO, those WebSphere MQ transactions are then committed. Then all MQ/204 subtasks are released.

A transfer to another subsystem is processed as a subsystem exit followed by a subsystem entry. Therefore, a transfer to another subsystem commits all active WebSphere MQ transactions and releases all MQ/204 substages. The first MQ/204 MQGET or MQPUT command in the new subsystem initiates a new WebSphere MQ transaction.

Controlling MQ subtask release

You can use the MQSUBREL parameter to control whether the MQ subtask is released at the end of request processing in subsystems with AUTOCOMMIT=N.

You can set the MQSUBREL parameter on User 0's CCAIN line to affect all users. Or, you can reset it for an individual user to affect only that specific user. At login or a user restart the MQSUBREL parameter is returned to the value specified by the User 0 CCAIN parameter line.

- A setting of 0 lets you retain the subtask and preserve pre-V7R1.0 behavior. Choose this setting when you expect the same user to use MQ services soon after ending a request.

- A setting of 1 lets you release the subtask for another user. Choose this setting when you expect that other users may be waiting for a free MQ subtask.

Grouping messages

MQ/204 takes advantage of the new features IBM introduced for WebSphere MQSeries, in particular the grouping of messages. The new fields in the MQMD Version 2 are Group ID and status, sequence number and flags. The Group ID, sequence number, and some of the flags determine whether a message is part of a group. Now Model 204 users can control the grouping of messages.

The other fields that apply to segmentation are not covered, because the segmentation options are not available for z/OS.

Supporting Java Message Service (JMS)

MQ/204 supports the predefined JMS protocol. MQ/204 now understands and can talk to Java through the messaging services. The JMS protocol, when sending MQ messages, expects to encode certain Java-specific fields into the additional header known as the MQRFH Version 2 (MQRFH2) header. The receiving mainframe application should expect to receive such a header, if it knows that the source of the messages is a JMS application.

Because the MQRFH2 header carries JMS-specific information, always include it in the message when you know that the receiving destination is a JMS application.

Conversely, omit the MQRFH2 when sending a message directly to a non-JMS application, because such an application does not expect an MQRFH2 in its WebSphere MQ message.

Environment requirements

Minimum is Model 204 V7R1.0.

Consulting IBM documentation

Rocket recommends that you keep these documents handy:

IBM Application Programming Reference.

IBM MQ manual, *Using Java*

Message groups

You can group messages together. Each message in a group has its own sequence number and shares the group ID number. The sequence number is assigned by system or by you, depending on whether you select the LOGICAL_ORDER or NOT_LOGICAL_ORDER option on your command or statement.

- The physical order of the messages in the group is the order in which the messages were written and that order is retained using the LOGICAL_ORDER option, so that the physical and logical order of the messages in the group match.
- You can set the logical order of the messages in the group using the NOT_LOGICAL_ORDER and the SEQUENCE option. The physical and logical order of the messages in the group may differ, as you assigned the sequence numbers.

In either case, the messages in the group are processed in ascending order and the highest sequence number in the group is called the last logical message.

When you use MQPUT to write messages, you have three options, which are discussed in turn:

- Group messages in logical order
- Group messages, but not in logical order
- Do not group messages

Messages grouped in logical order

Grouping messages in logical order is the more simple way of writing messages that are in a group.

1. On the first message you set the options:
   - GSTATUS='G'
   - LOGICAL_ORDER

   There is no need to set GROUPID or SEQUENCE, as the queue manager automatically assigns these values.
2. Continue writing messages to the queue with GSTATUS='G' and LOGICAL_ORDER options set until the last logical message
3. For the last message you set the options:
   - GSTATUS='L'.
   - LOGICAL_ORDER

Using this method, you can write messages to only one group at a time.

Message groups and MQPUT1

The LOGICAL_ORDER option is not allowed on the MQPUT1 statement, although the GSTATUS, SEQUENCE, and GROUPID options are allowed, so it is possible to write a message to a group using MQPUT1, but not in logical order.

Messages grouped, not in logical order

If you need to group messages, but cannot use the LOGICAL_ORDER option because either you need to write

- To two or more groups at the same time
- Messages where the physical sequence is not also the logical sequence (You are not writing the messages in ascending order of sequence number.)

Grouping messages, but not in logical order, is more complex to handle, but gives you more flexibility.

1. On the first message you must set the options:
   - GSTATUS to 'G', unless this is the last logical message in the group—the highest sequence number—in which case set GSTATUS to 'L'.
   - NOT_LOGICAL_ORDER
   - GROUPID to null or spaces
   - SEQUENCE to the sequence number of this message within the group, which must be greater than zero, although not necessarily one, because the messages can be out of sequence.
2. When this first MQPUT to the queue completes, you must save the GROUPID returned in the DESCRIPTOR field, because this is the Group ID that the queue manager has assigned to this group. You will set this for all subsequent messages in this group.
3. For all subsequent messages in the group set the options:
   - GSTATUS='G', unless this is the last logical message in the group—the highest sequence number—in which case set GSTATUS='L'.
   - GROUPID to the value returned from the first MQPUT (step 2)
   - NOT_LOGICAL_ORDER
   - SEQUENCE to the sequence of this message within the group.

If you write groups in this way, you can write to several groups at the same time. Be careful to save all the Group ID values from the first MQPUT for each group.

Messages not grouped

Messages not grouped was the only available behavior in MQ/204 V6R1.0 and earlier. You can maintain previous behavior by omitting GSTATUS or setting it to null or space (' '). The settings for SEQUENCE, GROUPID and LOGICAL_ORDER options are ignored.

Searching for messages using the MATCH options

You can retrieve messages that match criteria that you specify. The new options for the MQGET statement are:

```
MATCH_CORREL_ID or NOT_MATCH_CORREL_ID

MATCH_GROUP_ID

MATCH_MSG_ID or NOT_MATCH_MSG_ID

MATCH_MSG_SEQ_NUMBER

MATCH_MSG_TOKEN
```

Although these options are new, the behavior prior to MQ/204 V6R2.0 was to match on any supplied CORRELID and MSGID. To avoid upward compatibility issues, the MQ/204 V6R2.0 defaults are:

```
MATCH_CORREL_ID

MATCH_MSG_ID
```

There are no equivalent NOT_ options for the criteria of GROUPID, SEQUENCE, and MSGTOKEN, because simply omitting these MATCH_ options reverses their effect.

Before searching for and retrieving messages

To search and retrieve messages using the MATCH options, you must know the index type of the queue. The index type of the queue was set by the system manager when the queue was defined.

The Table 1-2 lists the index types, their purpose, and usage.

Table 1-2. Index types and purpose

Index type	Queue manager maintains	For queues
CORRELID	Correlation identifiers of the messages on the queue.	Where the application usually retrieves messages using the correlation identifier as the selection criterion on the MQGET call.
GROUPID	Group identifiers of the messages on the queue.	Where the application retrieves messages using the LOGICAL_ORDER option on the MQGET call.

Table 1-2. Index types and purpose (continued)

MSGID	Message identifiers of the messages on the queue.	Where the application usually retrieves messages using the message identifier as the selection criterion on the MQGET call.
MSGTOKEN	Message tokens of the messages on the queue for use with the workload manager (WLM) functions of z/OS.	That are WLM-managed queues; do not specify it for any other type of queue. Also, do not use this value for a queue where an application is not using the z/OS workload manager functions, but is retrieving messages using the message token as a selection criterion on the MQGET call.
No index	No index.	That are usually processed sequentially, that is, without using any selection criteria on the MQGET call.

Using index types with message groups

Table 1-3 and Table 1-4 on page 27 list the search criteria you can used when a queue is defined with particular index type(s). Since you can retrieve messages in logical order (Table 1-4 on page 27) or not logical order (Table 1-3), your MQGET statement can also include either the NOT_LOGICAL_ORDER or LOGICAL_ORDER option.

Table 1-3. Index types and NOT_LOGICAL_ORDER option

Selection criteria on MQGET call	Index type for nonshared queue	Index type for shared queue
None	Any	Any
Selection using one MATCH_ option		
MSGID	MSGID recommended	None or MSGID
CORRELID	CORRELID recommended	CORRELID required
GROUPID	GROUPID recommended	GROUPID required
Selection using two MATCH_ options		
MSGID + CORRELID	MSGID or CORRELID recommended	MSGID or CORRELID required
MSGID + GROUPID	MSGID or GROUPID recommended	Not supported
CORRELID + GROUPID	CORRELID or GROUPID recommended	Not supported

Table 1-3. Index types and NOT_LOGICAL_ORDER option (continued)

Selection criteria on MQGET call	Index type for nonshared queue	Index type for shared queue
Selection using three MATCH_ OPTIONS		
MSGID + CORRELID + GROUPID	MSGID or CORRELID or GROUPID recommended	Not supported
Selections using group-related criteria		
GROUPID + SEQUENCE	GROUPID required	GROUPID required
SEQUENCE (must be at least one)	GROUPID required	GROUPID required
Selection using MSGTOKEN		
For application use	Do not index by MSGTOKEN	Do not index by MSGTOKEN
For Work Load Management (WLM) use	MSGTOKEN required	Not supported

Table 1-4 lists the required index type when LOGICAL_ORDER is specified on a shared or not-shared queue.

Table 1-4. Index type and LOGICAL_ORDER option

Selection criteria on MQGET call	Index type for non-shared queue	Index type for shared queue
None	GROUPID required	GROUPID required
Selection using one MATCH_ option		
MSGID	GROUPID required	Not supported
CORRELID	GROUPID required	Not supported
GROUPID	GROUPID required	GROUPID required
Selection using two MATCH_ options		
MSGID + CORRELID	GROUPID required	Not supported
MSGID + GROUPID	GROUPID required	Not supported
CORRELID + GROUPID	GROUPID required	Not supported
Selection using three MATCH_ options		

Table 1-4. Index type and LOGICAL_ORDER option (continued)

Selection criteria on MQGET call	Index type for non-shared queue	Index type for shared queue
MSGID + CORRELID + GROUPID	GROUPID required	Not supported

Retrieving messages not grouped

If there is no current group or logical message, only messages that have SEQUENCE=1 are eligible for return. In this situation, one or more of the following match options can be used to select which of the eligible messages is the one actually returned:

- MATCH_CORREL_ID
- MATCH_GROUP_ID
- MATCH_MSG_ID

If LOGICAL_ORDER is specified, and there is a current group, only the next message in the group is eligible for return, and this cannot be altered by specifying MATCH_ options.

Match options which are not applicable can still be specified, but the value of the relevant field must match the value of the corresponding field in the message to be returned.

One or more of the following match options can be specified.

Table 1-5. Using the MATCH options

MATCH_ option	Retrieves message with specified	Message retrieved must have
CORREL_ID	Correlation identifier	Correlation identifier that matches the value of CORRELID, as well as any other matches that may apply, such as message identifier. If NOT_MATCH_ option is specified, the CORRELID field is ignored and any correlation identifier will match.
GROUP_ID	Group identifier	Group identifier that matches the value of GROUPID, as well as any other matches that may apply, such as correlation identifier.

Table 1-5. Using the MATCH options (continued)

MATCH_ option	Retrieves message with specified	Message retrieved must have
MSG_ID	Message identifier	Message identifier that matches the value of MSGID, as well as any other matches that may apply, such as correlation identifier. If NOT_MATCH_ option is specified, MSGID is ignored and any message identifier will match.
MSG_SEQ_NUMBER	Message sequence number	Message sequence number that matches the value of the SEQUENCE field, as well as any other matches, such as group identifier.
MSG_TOKEN	Message token	Message token that matches the value of MSGTOKEN—only for queues that have an index type of MSGTOKEN. You cannot specify other match options with MATCH_MSG_TOKEN.

BROWSE options

For MQ/204 after V6R1.0, there are new options for browsing a queue.

BROWSE_MSG_UNDER_CURSOR, a new option on the MQGET statement, lets you reread the same message. You can combine this option with the LOCK or UNLOCK options.

- LOCK option lets you lock a message so that another thread cannot browse it. The lock is automatically released at the end of the unit of work, even if there is no explicit UNLOCK. The LOCK option is also valid with BROWSE_FIRST and BROWSE_NEXT.
- UNLOCK option causes the message to be unlocked without retrieving the message.
- Omitting both LOCK and UNLOCK options unlocks the message, but also retrieves it again.

  The following sequence of calls is an example:

```
MQGET BROWSE_FIRST
MQGET BROWSE_NEXT
                  /*until the required message is found
MQGET BROWSE_MSG_UNDER_CURSOR LOCK
              /* to reread the last message and lock it
```

```
MQGET BROWSE_MSG_UNDER_CURSOR UNLOCK
/* unlock the same message without retrieving it again
```

Special handling options

ALL_MSGS_AVAILABLE option

You can specify the new ALL_MSGS_ AVAILABLE option on the MQGET statement, and you can set it as a default for the queue by specifying ALL_MSGS_AVAILABLE on the DEFINE QUEUE command or MODIFY QUEUE statement. The default setting is NOT_ALL_MSGS_AVAILABLE.

If ALL_MSGS_AVAILABLE is specified, a message that is part of a group cannot be retrieved from the queue unless all messages in the group are available for retrieval.

MSGTOKEN option

The 16-byte message token, like the message ID, identifies a message as unique. However, unlike the message ID, the message token is not passed between queue managers, so a message token changes as it passes from one queue manager to another queue manager.

You can retrieve the MSGTOKEN %variable option on a MQGET statement. It is also an input field on the MQGET statement, if you also specified the MATCH_MSG_TOKEN option.

NEW_CORREL_ID option

The NEW_CORREL_ID option on the MQPUT and MQPUT1 statements tells the queue manager to generate a new correlation ID for the message. The CORRELID option should not be specified with this field, because the queue manager generates a unique correlation identifier, rather than taking it from the CORRELID field.

SYNCPOINT_IF_PERSISTENT option

The SYNCPOINT_IF_PERSISTENT option is like the SYNCPOINT option, although the request is under transaction control (affected by MQCMIT and MQBACK statements) only if the message is PERSISTENT. This option is valid only on the MQGET statement.

Examples of writing messages and browsing groups

Example of writing messages to a group in logical order

The following example writes messages to a group in logical order. Note that it is unnecessary to specify SEQUENCE or GROUPID.

```
%QNAME IS STRING LEN 8
%MSG2  IS STRING LEN 100
%MSG1  IS STRING LEN 100
%MSGID IS STRING LEN 24
%COUNT IS FIXED
%LO    = 'LOGICAL_ORDER'
%MSG1  = 'PUTS 4 MESSAGES IN A LOGICALLY ORDERED GROUP'
%COUNT = 0
%GSTAT = 'G'    /? group status for first four messages?/
OPEN QUEUE %QNAME   OUTPUT
REPEAT 3 TIMES
  %COUNT = %COUNT + 1
  %MSG2  = %MSG1 WITH %COUNT
  MQPUT %MSG2 ON %QNAME   GSTATUS %GSTAT LOGICAL_ORDER
  CALL PRINT.STATUS               /? test return codes ?/
END REPEAT
%GSTAT = 'L'     /? group status for last message only ?/
%COUNT = %COUNT + 1
%MSG2 = %MSG1 WITH %COUNT
MQPUT %MSG2 ON %QNAME   GSTATUS %GSTAT  ?%LO
          /? logical order specified as runtime option ?/
```

Example of writing messages to a group out of sequence

The following example writes messages to a group out of sequence. Notice that we specify NOT_LOGICAL_ORDER, and take care to set SEQUENCE, GSTATUS, and GROUPID correctly.

```
%MSG1  = '5 MESSAGES IN A GROUP OUT OF SEQUENCE'
%COUNT = 0
*             imbed the image definition for the message descriptor
I MQMDV2
PREPARE IMAGE MQMD
%NLO   = 'NOT_LOGICAL_ORDER'
*                for the first physical PUT, the group must be null
%GRP   = ' '
*                    GSTATUS must be L for the last logical message
%GSTAT = 'L'                     /? group status for SEQUENCE = 5 ?/
OPEN QUEUE %QNAME   OUTPUT
%SEQ  = 5
*              we can write the message with sequence number 5 first
```

```
REPEAT 4 TIMES
  %COUNT = %COUNT + 1
  %MSG2  = %MSG1 WITH %COUNT
  *                          specify DESCRIPTOR to retrieve the GROUPID
  MQPUT %MSG2 ON %QNAME  GSTATUS %GSTAT -
   SEQUENCE %SEQ       /? this group in reverse order of sequence ?/ -
   NOT_LOGICAL_ORDER -
   GROUPID %GRP     /? null for first PUT, then generated GROUPID ?/ -
   DESCRIPTOR MQMD          /? needed to retrieve generated GROUPID ?/

   %SEQ   = %SEQ - 1           /? next PUT will have lower SEQUENCE ?/
   CALL PRINT.STATUS                /? always check the return code ?/
   *                          GSTATUS must be G for all other messages
   %GSTAT = 'G'                 /? group status for SEQUENCE 1 to 4 ?/
   %GRP   = $MQMD:GROUPID         /? retrieve the generated GROUPID ?/
  PRINT 'Generated GROUPID IS: ' WITH %GRP
END REPEAT
%COUNT = %COUNT + 1
%MSG2  = %MSG1 WITH %COUNT
MQPUT %MSG2 ON %QNAME   GSTATUS %GSTAT  -
 SEQUENCE %SEQ         /? last PUT has SEQUENCE 1 in this example ?/ -
 ?%NLO GROUPID %GRP          /? use generated GROUPID for last time ?/
CALL PRINT.STATUS
```

Browsing a group of messages

This example browses a group of messages. In this case we are using the option MATCH_GROUP_ID to select messages in a certain group, and LOGICAL_ORDER to ensure that we retrieve them in sequence number order. The index type for this queue must therefore be GROUPID.

```
%QN IS STRING LEN 8
%MG = 'MATCH_GROUP_ID'
%GI = 'GROUPID %GRP'
%LO = 'LOGICAL_ORDER'
INCLUDE MQMD         /? image definition of V2 message descriptor ?/
%COUNT IS FIXED
%TEXT IS STRING LEN 255
%GRP IS STRING LEN 24
%MSGID IS STRING LEN 24
%MSGTOKEN IS STRING LEN 16
%CORRELID IS STRING LEN 24
%GSTAT    IS STRING LEN 1
%SEQ IS FIXED
*                    you can specify LOGICAL_ORDER on OPEN or MODIFY
OPEN QUEUE %QN BROWSE %LO
CALL PRINT.STATUS                 /? always check the return code ?/
%COUNT = 0
MQGET %TEXT FROM %QN   NO_WAIT   -
```

```
 BROWSE_FIRST                -
 DESCRIPTOR MQMD                              /? retrieve group ID ?/
CALL PRINT.STATUS
%GRP      = %MQMD:GROUPID               /? save these values from ?/
%SEQ      = %MQMD:MSGSEQNO              /? the message descriptor ?/
%MSGID    = %MQMD:MSGID
%CORRELID = %MQMD:CORRELID
*                 NOW READ THE WHOLE QUEUE USING THE 'MATCH' KEYWORD
MQGET %TEXT FROM %QN      NO_WAIT   -
 BROWSE_FIRST ?%MG       /? MATCH_GROUP_ID can be a runtime var ?/ -
 ?%GI             /? GROUPID %GRP is another run-time variable ?/  -
 DESCRIPTOR MQMD
CALL PRINT.STATUS
%COUNT = %COUNT + 1
READ.NEXT:
MQGET %TEXT FROM %QN      NO_WAIT  -
 ?%MG ?%GI              -
 BROWSE_NEXT   -
 DESCRIPTOR MQMD GSTATUS %GSTAT
CALL PRINT.STATUS
IF $STATUS = 23 THEN
   JUMP TO END.LOOP       /? end if no more messages in the group ?/
END IF
IF $STATUS = 12 THEN
   JUMP TO END.LOOP                      /? test for other errors ?/
END IF
%COUNT = %COUNT + 1                /? count messages in the group ?/
*     It would also be possible to test for %GSTAT = 'L' to test for
*                                     the end of the loop, as here:
IF %GSTAT = 'L'                      /? last message in the group ?/
 JUMP TO END.LOOP         /? end if no more messages in the group ?/
END IF
JUMP TO READ.NEXT
END.LOOP:
*   $STATUSD = 2247 if the Queue has wrong index type
IF $STATUSD = 2247 THEN
   AUDIT 'WRONG TYPE OF INDEX FOR MATCH'
ELSE
   AUDIT ' Messages in the group: ' WITH %COUNT
END IF
CLOSE QUEUE %QN
```

Supporting Java messages with the RFH2 keyword

The RFH2=(*image* | BUFFER) option of the MQGET and MQPUT statements support Java messages and the MQRFH2 header. The following example includes SOUL statements using the RFH2 keyword.

```
WRITE IMAGE MQRFH2 ON BUFFER POSITION 1
```

```
*             the RFH2 header is now in the buffer
MQPUT DATA1  ON %QNAME   GSTATUS %GSTAT -
 LOGICAL_ORDER RFH2 BUFFER -
 GROUPID %GRP  DESCRIPTOR MQMD
*  The following MQGET reads the RFH2 header into the
*  image called MQRFH2, and the data into the %variable
*  called %TEXT
MQGET %TEXT FROM %QNAME NO_WAIT    -
 DESCRIPTOR MQMD -
 RFH2 MQRFH2
```

See “MQGET statement” on page 68 and “MQPUT statement” on page 75 for syntax layout and more details.

Updating the Version 2 message descriptor (MQMD V2)

The image definition for the message descriptor, which you can retrieve using the DESCRIPTOR option, is updated to include the new fields in MQMD Version 2. The following sample image can be specified as target of the WebSphere MQ MQGET DESCRIPTOR option. It matches the layout of the WebSphere MQ MQMD data structure that is described in the *WebSphere MQ Application Programming Reference.*

```
 IMAGE MQMD
* THIS IS THE VERSION 2 MQMD
STRUCID     IS STRING LEN 4
* TYPE OF STRUCTURE
VERSION     IS BINARY LEN 4
* VERSION NUMBER OF THE STRUCTURE
REPORT      IS BINARY LEN 4
* OPTIONS FOR REPORT MESSAGES
MSGTYPE     IS BINARY LEN 4
* THE TYPE OF MESSAGE
* (1=REQUEST, 2=REPLY, 4=REPORT, 8=DATAGRAM)
EXPIRY      IS BINARY LEN 4
* MESSAGE LIFETIME
FEEDBACK    IS BINARY LEN 4
* FEEDBACK CODE
ENCODING    IS BINARY LEN 4
* DATA ENCODING
CODECHARSETID  IS BINARY LEN 4
* CODED CHARACTER SET IDENTIFIER
FORMAT         IS STRING LEN 8
* FORMAT NAME
PRIORITY       IS BINARY LEN 4
* MESSAGE PRIORITY
PERSISTENCE    IS BINARY LEN 4
* MESSAGE PERSISTENCE
MSGID          IS STRING LEN 24
* MESSAGE IDENTIFIER
```

```
CORRELID       IS STRING LEN 24
* CORRELATION IDENTIFIER
BACKOUTCOUNT   IS BINARY LEN 4
* BACKOUT COUNTER
REPLYTOQ       IS STRING LEN 48
* NAME OF REPLY QUEUE FOR REQUESTS
REPLYTOMGR     IS STRING LEN 48
* NAME OF REPLY QUEUE MANAGER FOR REQUESTS
USERIDENTIFIER IS STRING LEN 12
* USER IDENTIFIER
ACCOUNTINGTOKEN  IS STRING LEN 32
* ACCOUNTING TOKEN
APPLIDENTITYDATA IS STRING LEN 32
* APPL DATA RELATING TO IDENTITY
PUTAPPLTYPE      IS BINARY LEN 4
* TYPE OF APPLICATION THAT PUT THE MSG
PUTAPPLNAME      IS STRING LEN 28
* ID OF APPLICATION THAT PUT THE MESSAGE
PUTDATE          IS STRING LEN 8
* DATE WHEN MESSAGE WAS PUT
PUTTIME          IS STRING LEN 8
* TIME WHEN MESSAGE WAS PUT
APPLORIGINDATA   IS STRING LEN 4
* APPLICATION DATA RELATING TO ORIGIN
GROUPID          IS STRING LEN 24
* GROUP ID
MSGSEQNO         IS BINARY LEN 4
* MESSAGE SEQUENCE NUMBER WITHIN GROUP
OFFSET           IS BINARY LEN 4
* OFFSET OF SEGMENT WITHIN MESSAGE
MSGFLAGS         IS BINARY LEN 4
* MESSAGE FLAGS
ORGLEN           IS BINARY LEN 4
* ORIGINAL LENGTH
END IMAGE
```

MQRFH2 image format

You can use the following image when there are no variable length fields.

```
IMAGE MQRFH2
* THIS IS THE RFH2 HEADER FOR JAVA
STRUCID     IS STRING LEN 4
* TYPE OF STRUCTURE'RFH '
VERSION     IS BINARY LEN 4
* VERSION NUMBER OF THE STRUCTURE - 2
LENGTH      IS BINARY LEN 4
* LENGTH OF THE STRUCTURE (36)
ENCODING    IS BINARY LEN 4
```

```
* THE TYPE OF ENCODING
CHARSET     IS BINARY LEN 4
* CODEDCHARSETID OF DATA
FORMAT      IS STRING LEN 8
* FORMAT OF DATA FOLLOWING RFH2
FLAGS       IS BINARY LEN 4
* FLAGS - SET TO 0
NAMEVALUECCSID IS BINARY LEN 4
* 1208, 1200, 13488, OR 17584
* CODEDCHARSETID OF VARIABLE PART
END IMAGE
```

MQMD version compatibility

MQ messages written with the MQ Message Descriptor (MQMD) Version 2 cannot be read by Model 204 V6R1.0, or earlier, Onlines. However, MQ messages written with MQMD Version 1 can by read by Model 204 V6R3.0 Onlines.

The MQMD is a part of the MQ message. It is referred to as the message header. An Online that uses Version 1 of the MQMD header, such as MQ/204 V6R1.0, cannot read messages from a Version 2 MQMD header, due to additional fields in the message header. However, an Online that uses Version 2 of the MQMD header, such as MQ/204 V6R3.0, can read messages from a Version 1 MQMD header, because the Version 2 header new fields provide default values.

Programming suggestions

Messages that cause errors

Occasionally a message taken from a queue causes an error; that is, the error is in the message. When you back out and try to retrieve the message again, the error recurs.

To break this back out loop, set the MARK_SKIP_BACKOUT option in a subsequent MQGET statement. When this option is set, the message taken from the queue under syncpoint control is not placed back on the queue when a user or the Online issues an MQBACK statement.

Removing messages that do not convert

By default WebSphere MQ tries to perform a data conversion, for example from EBCDIC to ASCII. If you issue an MQGET statement for a message that fails to convert, the message becomes stuck on the queue.

To get the message off the queue, reissue the MQGET statement with the NO_CONVERT option. The message can now be taken off the queue, because the MQGET processing completes successfully, as in the following example:

```
MQGET
IF $STATUS=12 $STATUSD=8
  THEN
  MQGET… NO_CONVERT
  %CONVERTED_FLAG=0
```

Saving a permanent local dynamic queue name

If your application creates permanent local dynamic queues that make use of queue name patterns, you might want to save the name of the generated permanent local dynamic queue in case the run comes down unexpectedly.

To save a permanent local dynamic queue name:

1. Issue an OPEN QUEUE statement with the MODEL and DYNAMICQNAME options that specifies the permanent local dynamic queue that you want to save.
2. Use the $MQ_QUEUENAME function to obtain the full external name of the queue.
3. Store the full external queue name in a procedure that your run includes when it comes up. Or, you can store it in a database so a procedure that your run includes when it comes up can dynamically generate the DEFINE QUEUE command to identify the queue.

When you decide to delete the queue, remove the queue name from the procedure or database.

Working with logically deleted queues

On z/OS, until the last message and request are closed, the queue is logically deleted but still exists. For example, you can still display the queue. However, you cannot retrieve messages or put messages on a logically deleted queue. During this state, any attempt to create a new queue, either local dynamic or predefined, with the same name fails. In the case of a local dynamic queue, the OPEN QUEUE statement fails with the reason code QRC_NAME_IN_USE. This is true for the application that logically deleted the queue, as well as for other applications.

After the last reference to the queue is closed, the queue is physically deleted; you can now create a new queue with the same name. However, in the case of a temporary local dynamic queue, if any corresponding unresolved units of work are outstanding, the queue can be physically deleted only when the application, which is holding the queue open, terminates.

Occasionally a logically deleted, permanent local dynamic queue has uncommitted updates. In this case, the queue is physically deleted only after resolving the corresponding units of work, as well as closing all the handles.

Tuning MQ/204

Consider the following ways to improve the performance of MQ/204:

- Set the initial value of UBUFSZ accurately.
- Make sure that the number of MQ/204 subtasks is optimal. When this setting is too low, the elapsed time for individual users might be higher than necessary, because they must wait for a free subtask. Too many subtasks may cause additional z/OS expenses to control subtasks.
- WebSphere MQ performance can be affected by message size. WebSphere MQ performance tends to be relatively better when large message sizes, above 4K, are used.
- Tune WebSphere MQ according to the recommendations of WebSphere MQ system managers and documentation.

Greenwich Mean Time and MQPUT, MQPUT1, and MQGET time

WebSphere MQ uses Greenwich Mean Time (GMT) when storing messages, because it is a transport tool that can pass messages across multiple time zones. WebSphere MQ does not have a parameter that you can set to use local time rather than Greenwich Mean Time.

If your site has chosen to reset the clock on your mainframe from Greenwich Mean Time to your local time, rather than keeping the mainframe on Greenwich Mean Time and calculating the local time offset, you might notice that the time-stamp for MQPUT statements is offset from the time-stamps of WebSphere MQ on PCs and other mainframes in your network.

See “Applying date and time-stamps to messages” on page 79 for a more detailed discussion.

Increase in STBL for MQ/204 sites

In Model 204 V6R3.0 and later, the increased STBL requirement is because MQ control blocks are kept in STBL, and they have increased in size by several hundred bytes. The increase depends on how many MQPUT and MQGET statements are compiled in any one transaction, so it will be the high water mark for the largest SOUL compilations.

MQ/204 sample application

The following annotated application sends and retrieves two messages.

Although omitted from the application to save space, do the following:

- Issue a SETGRC command to check that the DEFINE commands and START command work.
- Check $STATUS and $STATUSD after opening and closing the queues.

```
B
%X=$SETG('TIMES',2)
END

* The following commands define a queue manager, then a
queue,
* and launch the queue manager. You must define a queue
manager
* before you define a queue, because a queue cannot exist
(or be
* defined) without a queue manager.

DEFINE QM CCAQM1   WITH SCOPE=SYSTEM QMNAME=CSQ1
DEFINE Q  CCAQM1Q1 WITH SCOPE=SYSTEM QM=CCAQM1 -
QNAME=DVCCA.TEST2.PS.Q01
START QM CCAQM1

*

BEGIN
VARIABLES ARE UNDEFINED
%MSGID IS STRING LEN 24
%MSGID=$SUBSTR($USER,1,8) WITH $SUBSTR($DATE(2,''),2)
WITH -
$SUBSTR($TIME,1,2) WITH -
       $SUBSTR($TIME,4,2) WITH -
       $SUBSTR($TIME,7,2)
 PRINT 'MESSAGE ID ="' %MSGID '"'
 %X IS STRING LEN 4
 %X=$SETG('MSGID',%MSGID)
 %TEXT IS STRING LEN 255
 %INDEX IS FLOAT

* Within the SUBROUTINE…END SUBROUTINE statement, the
* PRINT.STATUS subroutine checks whether $STATUS or $STA-
TUSD is
* other than zero; if so, it prints the values. PRINT.STA-
TUS
* subroutine is called several times during this applica-
tion to
* provide queue information.

 SUBROUTINE PRINT.STATUS
 IF $STATUS NE 0 OR $STATUSD NE 0 THEN
  PRINT '$STATUS/$STATUSD=' WITH $STATUS WITH '/' WITH
$STATUSD
  SKIP 1 LINE
 END IF
RETURN
END SUBROUTINE
```

```
* The following code opens a queue (CCAQM1Q1) and calls
the
* PRINT.STATUS subroutine. The FOR loop creates a message
and
* puts it on the queue (MQPUT statement).

OPEN QUEUE CCAQM1Q1 OUTPUT
CALL PRINT.STATUS
PRINT 'ADDING ?&TIMES MESSAGES'
FOR %INDEX FROM 1 TO ?&TIMES BY 1
*PRINT 'ABOUT TO ADD FOLLOWING MESSAGE TO THE QUEUE'
 %TEXT='!! THAT''S AMAZING AT ' WITH $TIME WITH ' !!' -
   WITH %INDEX WITH ' ' WITH %MSGID
 PRINT %TEXT
 MQPUT %TEXT ON CCAQM1Q1 MSGID=%MSGID
 CALL PRINT.STATUS
END FOR
CLOSE QUEUE CCAQM1Q1
CALL PRINT.STATUS
END

*

BEGIN
VARIABLES ARE UNDEFINED
%TEXT IS STRING LEN 255
%MSGID IS STRING LEN 24
%MSGID=$GETG('MSGID')
SUBROUTINE PRINT.STATUS
IF $STATUS NE 0 OR $STATUSD NE 0 THEN
 PRINT '$STATUS/$STATUSD=' WITH $STATUS WITH '/' WITH
$STATUSD
 SKIP 1 LINE
END IF
RETURN
END SUBROUTINE
OPEN QUEUE CCAQM1Q1
CALL PRINT.STATUS
PRINT 'ATTEMPT TO RETRIEVE ?&TIMES MESSAGES'

* The following REPEAT loop retrieves the messages.

REPEAT ?&TIMES TIMES
 MQGET %TEXT FROM CCAQM1Q1 NO_WAIT MSGID=%MSGID
 CALL PRINT.STATUS
 PRINT $TIME WITH ' MQGET: ' WITH %TEXT
END REPEAT
CLOSE QUEUE CCAQM1Q1
CALL PRINT.STATUS
END
```

Sample output

```
MESSAGE ID="00003980501121739"
ADDING 2 MESSAGES
!! THAT'S AMAZING AT 12:17:39 !!1 00003980501121739
!! THAT'S AMAZING AT 12:17:40 !!2 00003980501121739
ATTEMPT TO RETRIEVE 2 MESSAGES
12:17:40 MQGET: !! THAT'S AMAZING AT 12:17:39 !!1
00003980501121739
12:17:40 MQGET: !! THAT'S AMAZING AT 12:17:40 !!2
00003980501121739
```

MQ/204 restrictions

The WebSphere MQ API supports a set of features that permits many functions to be performed with queues. The MQ/204 interface supports many, but not all these features. The following features are *not* supported:

- MQINQ and MQSET, query and set attributes of objects.
- Direct access to the MQCONN and MQDISC calls or connection handles.

  Connections to queue managers are handled internally by OPEN QUEUE and CLOSE QUEUE statements. (See the WebSphere MQ documentation for a discussion of connection handles.)
- Direct access to the WebSphere MQ control blocks (get message options, put message options, and so on), except for the message descriptors on MQGET statements.

  MQ/204 is a higher level keyword based interface, which limits functions to those within the keyword interface.
- WebSphere MQ API method of asynchronously fetching messages (the WebSphere MQ option MQGMO_SET_SIGNAL).

2

Monitoring and Troubleshooting

This chapter discusses:

- Installation considerations particular to MQ/204
- Tools available within Model 204 to develop and debug MQ/204 applications

Error handling with $STATUS and $STATUSD

$STATUS is the primary return code and $STATUSD is an ancillary or secondary return code. Certain $STATUS return codes automatically generate a $STATUSD return code. For other $STATUS return codes, the value of $STATUSD is 0.

$STATUS return codes

Table 2-1 lists the $STATUS return code values and text that is used for all MQ/204 SOUL statements. See Chapter 4 for a detailed description of each statement.

When an error can be returned by more than one statement, it has the same $STATUS value for all statements that can raise that error. This practice allows

you to write a generic SOUL routine that is an error handler for all MQ/204 statements.

Table 2-1. $STATUS values for MQ/204 errors

$STATUS Value	Meaning	Statement
1	Queue or queue manager was not found.	CLOSE QUEUE MODIFY QUEUE MQBACK MQCMIT MQGET MQPUT MQPUT1 OPEN QUEUE
2	Reply queue not found.	MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
3	Queue manager is not started.	MQPUT1 OPEN QUEUE
4	No WebSphere MQ interface subtasks are available.	MQPUT1 OPEN QUEUE
5	WebSphere MQ A PI level MQCONN failed. $STATUSD is set to the WebSphere MQ reason code.	MQPUT1 OPEN QUEUE
6	Queue manager has been stopped.	MQPUT1 OPEN QUEUE
7	Queue is already open.	OPEN QUEUE
8	Storage allocation error.	MODIFY MQ_BUFFER MQBACK MQCMIT MQGET MQPUT MQPUT1 OPEN QUEUE READ IMAGE WRITE IMAGE
9	WebSphere MQ A PI level MQOPEN failed. $STATUSD is set to the WebSphere MQ reason code.	OPEN QUEUE

Table 2-1. $STATUS values for MQ/204 errors (continued)

$STATUS Value	Meaning	Statement
10	%Variable substitution error, consult $STATUSD for more details. (See Table 2-3 on page 49.)	MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
11	Specified queue is not open.	CLOSE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1
12	WebSphere MQ A PI level MQGET failed. $STATUSD is set to the WebSphere MQ reason code.	MQGET
13	Browse was attempted on MQGET, but the queue was not open for browsing.	MQGET
14	Data truncation occurred on an MQGET. The target image or variable for the data area is too small.	MQGET
15	WebSphere MQ A PI level MQPUT failed. $STATUSD is set to the WebSphere MQ reason code.	MQPUT
16	WebSphere MQ A PI level MQPUT1 failed. $STATUSD is set to the WebSphere MQ reason code.	MQPUT1
17	MQPUT was attempted, and the queue was not open for output.	MQPUT
18	WebSphere MQ A PI level MQCMIT failed. $STATUSD is set to the WebSphere MQ reason code.	MQCMIT
19	WebSphere MQ API level MQBACK failed. $STATUSD is set to the WebSphere MQ reason code.	MQBACK
20	WebSphere MQ API level MQCLOSE failed. $STATUSD is set to the WebSphere MQ reason code.	CLOSE QUEUE
21	User is not connected to the queue manager specified.	MQBACK MQCMIT

Table 2-1. $STATUS values for MQ/204 errors (continued)

$STATUS Value	Meaning	Statement
22	Destructive get operation attempted, but the queue was not open for input.	MQGET
23	MQGET was issued, but no message was available.	MQGET
24	MQGET was issued with MSG_UNDER_CURSOR option, but the queue was not open for both input and browse.	MQGET
25	Data does not fit in the buffer and has been truncated.	MQPUT MQPUT1 WRITE
26	No message is in the message buffer, so no message was put.	MQPUT MQPUT1
27	Statement tried to reference a remote queue, but MQDEQMAN was not set.	CLOSE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
28	An OPEN QUEUE statement was issued for a remote queue, but the mode option specified was not OUTPUT.	OPEN QUEUE
30	Context queue entity was not found.	MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
31	Context queue not specified or inherited when it was needed, because a PASS option was specified.	MQPUT MQPUT1 OPEN QUEUE
32	Context queue not open for INPUT and SAVE_ALL_CONTEXT.	MQPUT MQPUT1
33	On PASS_USER_CONTEXT there was no user ID to pass.	MQPUT1 OPEN QUEUE
34	Invalid value of SIZE on MODIFY MQ_BUFFER.	MODIFY MQ_BUFFER

Table 2-1. $STATUS values for MQ/204 errors (continued)

$STATUS Value	Meaning	Statement
35	LIBUFF too small while parsing runtime options.	CLOSE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
36	Invalid or unrecognized option found while parsing runtime options.	CLOSE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
37	Unexpected end of option string while parsing runtime options.	CLOSE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
38	Duplicate options found while parsing runtime options.	CLOSE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
39	While parsing runtime options, *keyword*=%*variable* found, but %*variable* is undefined, $STATUSD set. (See Table 2-3 on page 49.)	CLOSE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
40	While parsing runtime options, *keyword*=*value* found, but *value* invalid, $STATUSD set. (See Table 2-3 on page 49.)	CLOSE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE

Table 2-1. $STATUS values for MQ/204 errors (continued)

$STATUS Value	Meaning	Statement
41	While parsing runtime options, mutually exclusive options found.	CLOSE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
42	While parsing runtime options, invalid combination of options found, $STATUSD set.	CLOSE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
43	Local dynamic queue, but null dynamic queue name.	OPEN QUEUE
44	Local dynamic queue, but null model queue name.	OPEN QUEUE
45	Local dynamic queue, but no model queue entity.	OPEN QUEUE
46	Local dynamic queue, but dynamic queue already exists.	OPEN QUEUE
47	RFH2 expected but not received.	
48	RFH2 imaged specified, but too small	
49	Error moving data to image	
50	Size error with multiple images on PUT(1)	
51	GCORE error saving data with PUT(1) and RFH2	
52	BUFLEN specified too small for RFH2 header	

%Variable substitution errors

Table 2-2 lists $STATUS values that indicate a %variable substitution error, when the option was evaluated, and how the option was initially specified. You

specified the value of an option as a %variable. When the statement was executed, the %variable contained an invalid value for the option in question.

Table 2-2. $STATUS values that indicate a %variable substitution error

$STATUS value	Evaluated at	Option initially specified as…
10	Compile time, as a %variable	%Variable
39	Runtime, as a ?%variable	%Variable
40	Runtime, as a ?%variable	Literal

To find out which $STATUS option had a problem, check the $STATUSD values, as listed in Table 2-3 on page 49.

$STATUSD return codes

On all $STATUS return codes that indicate a failure of a WebSphere MQ API level call, the value of $STATUSD is set to the WebSphere MQ reason code. The values and meanings of the reason codes are described in applicable IBM WebSphere MQ documents. For reason codes, see *WebSphere MQ for z/OS Messages and Codes V5.3.1,* Appendix A.

To find out which option had a problem, check $STATUSD to determine which option, as listed in Table 2-3.

Table 2-3. Identifying invalid options by $STATUSD value

$STATUSD value	Indicates invalid value for…
1	MSGTYPE
2	PRIORITY
3	REPLY_QUEUE
4	MSGID
5	CORRELID
6	EXPIRY
7	FEEDBACK
8	FORMAT
9	WAIT_TIME
10	CONTEXT
11	PUTAPPLTYPE
12	USERIDENTIFIER
13	ACCOUNTINGTOKEN

Table 2-3. Identifying invalid options by $STATUSD value (continued)

$STATUSD value	Indicates invalid value for…
14	APPLIDENTITYDATA
15	PUTAPPLNAME
16	PUTDATE
17	PUTTIME
18	APPLORIGINDATA
19	MAXLEN
20	POSITION
21	BUFLEN
22	DESCRIPTOR
23	REPORT
24	MSGLEN
25	REPLY_QMGR
26	DYNAMICQNAME
27	MODEL
28	GROUPID
29	GSTATUS
30	Options that save context require queue to be open with an input option
31	Options that allow and pass context require queue to be open for output
32	APPLORIGINDATA, PUTAPPLNAME, PUTAPPLTYPE, PUTDATE, and PUTTIME options that require the SET_ALL_CONTEXT option
33	USERIDENTIFIER, ACCOUNTINGTOKEN, and APPLIDENTITYDATA options that require SET_IDENTITY_CONTEXT or SET_ALL_CONTEXT
34	DYNAMICQNAME and MODEL options that must be specified together
35	MSGTOKEN
36	SEQUENCE

For all other values of $STATUS, $STATUSD is set to 0.

Debugging aid

Information recorded in the audit trail aids in debugging MQ/204 applications.

Audit trail

Each MQ/204 statement executed is logged to the audit trail as RK lines. Each RK line contains the following information:

- Current user ID
- Statement being executed, for example, OPEN QUEUE, MQGET, and so on.
- Queue manager name
- If the call is queue specific, the queue name
- Completion code and reason code returned by the WebSphere MQ API call

For OPEN QUEUE statements, an additional RK line is logged including:

- External queue manager name
- External queue name

For MQCONN and MQDISC calls, which are made on behalf of the user but are not strictly under the control of the SOUL program, a different RK line is logged containing the following information:

- Current user ID
- Function being performed, MQCONN or MQDISC
- Queue manager name
- External queue manager name
- Completion code and reason code returned by the WebSphere MQ API call
- Address of the associated subtask control block

Each of the RK lines has its own message number, so that each can be controlled independently with Model 204 MSGCTL parameter.

Wait types and statistics

To ensure that the system manager has full information, wait types identify waits that a user thread might perform when using MQ/204.

Wait type	Value	Used when a thread is waiting for …
MQAPI	41	WebSphere MQ operation to complete. The one exception is for the MQGWT wait type.

Wait type	Value	Used when a thread is waiting for …
MQGWT	42	WebSphere MQ MQGET operations, where the WAIT option has been specified.
MQTSK	40	Subtask to become available (the MQWAIT parameter must be nonzero to have this type of wait).

Measuring the throughput of the WebSphere MQ API

The following statistics attempt to measure the throughput of the WebSphere MQ API itself. To separate API execution time from time spent waiting within the API, MQGET calls with the WAIT option specified are measured separately. The following statistics measure all WebSphere MQ API calls except MQGET with the WAIT option.

Statistic name	Tracks…
MQAPICNT	Number of MQAPI waits performed. See “WebSphere MQ API wait types access control” on page 15.
MQAPITIM	Elapsed time spent in MQAPI waits. See “WebSphere MQ API wait types access control” on page 15.

Measuring MQGET calls with the WAIT options

The following statistics measure MQGET calls performed with the WAIT option and a finite WAIT_TIME value specified. MQGET calls with the WAIT option and WAIT_TIME=UNLIMITED specified are intentionally excluded from these measurements.

Statistic name	Tracks…
MQGWTCNT	Number of MQGWT waits performed for MQGET calls with the WAIT option and a finite WAIT_TIME value specified. See “WebSphere MQ API wait types access control” on page 15.
MQGWTSUC	Number of MQGET calls performed with the WAIT option and a finite WAIT_TIME specified that succeeded; for example, where a message was actually returned.
MQGWTTIM	Elapsed time spent in MQGWT waits done for MQGET calls with the WAIT option and a finite WAIT_TIME value specified. See “WebSphere MQ API wait types access control” on page 15.
MQGWTTSP	Total WAIT_TIME specified on all MQGET calls performed with the WAIT option and a finite WAIT_TIME value specified.

3

MQ/204 Command Reference

MQ/204 commands support the following tasks:

- DEFINE commands identify queue managers and queues with defaults for various get and put options.
- The system manager uses the BUMP, START, and STOP commands to control access to the queue managers.

If an MQ/204 command is issued but MQ/204 is not linked with Model 204, the following error message is issued:

```
M204.1039: FEATURE NOT LINKED IN
```

BUMP QUEUEMANAGER: Disconnecting queue manager users

Privileges System manager or User 0

Function Disconnects all users who have connections to the specified queue manager.

Syntax `BUMP {QUEUEMANAGER | QM} name`

Where *name* identifies a queue manager created with a DEFINE QUEUEMANAGER command.

Usage When you want to disconnect users from a queue manager, follow these steps:

1. Issue a STOP QUEUEMANAGER command to prevent additional access to the queue manager.
2. Issue a BUMP QUEUEMANAGER command. Any active WebSphere MQ

transactions for bumped users are backed out.

If the queue manager does not exist or has not been started, an error message is issued.

DEFINE QUEUE: Identifying a WebSphere MQ queue

Privileges System manager or User 0

Function Identifies an z/OS-WebSphere MQ queue in a queue manager where applications can put (MQPUT and MQPUT1) and get (MQGET) messages. The queue must have been previously defined by the WebSphere MQ administrator.

Syntax

```
DEFINE {QUEUE | Q} name [LIKE previousname] WITH

SCOPE=SYSTEM {QUEUEMANAGER | QM}=qmanentityname

[{QUEUENAME | QNAME}=externalqueuename] [options…]
```

Where

- *name* identifies the queue referred to in the Online. name must be:
  - Unique, or an error is issued.
  - 1-48 characters long and begin with a letter, followed by letters, numbers, periods, or underscores.
- QUEUEMANAGER (or QM) clause names the queue manager for the queue, and must refer to a queue manager previously defined by the WebSphere MQ administrator.

  Use the optional QUEUENAME (or QNAME) clause, if the external name of the queue (as defined to WebSphere MQ) differs from the name you refer to in the Online. External queue names can be up to 48 characters in length. The external queue name is validated when the queue is actually opened. If you omit QUEUENAME (or QNAME), the external name of the queue is assumed to be name.
- *options* are default characteristics that are used to manipulate the queues.
- ACCEPT_TRUNCATED_MSG or NO_ACCEPT_TRUNCATED_MSG (default)
- ALL_MSGS_AVAILABLE or NOT_ALL_MSGS_AVAILABLE (default)
- FAIL_IF_QUIESCING or NO_FAIL_IF_QUIESCING (default)
- CONTEXT
- EXPIRY
- FORMAT
- LOGICAL_ORDER or NOT_LOGICAL_ORDER (default)

- MSGTYPE
- PERSISTENT, NOT_PERSISTENT, or PERSISTENCE_AS_Q_DEF (default)
- PRIORITY or PRIORITY_AS_Q_DEF (default)
- REPLY_QMGR
- REPLY_QUEUE
- REPORT
- SYNCPOINT (default) or NO_SYNCPOINT
- WAIT or NO_WAIT (default)
- WAIT_TIME

Usage

The queue definition has system scope.

Note: Remote queues do not require a DEFINE QUEUE command. Local dynamic queues require a DEFINE QUEUE command only if they were not created by MQ/204 in the current Online job.

DEFINE QUEUE can be omitted if the MQDEQMAN keyword is supplied. (See “Parameters and task management” on page 12 for a description of MQDEQMAN.)

When a queue is known to the default queue manager, as specified by the MQDEQMAN keyword, it is possible to omit an explicit DEFINE QUEUE command. Instead MQPUT and MQGET processing can specify the external queue name.

If MQDEQMAN is not specified or the supplied queue name is not valid, the MQPUT or MQGET statement is rejected with the message:

```
M204.0630: IMPLICIT DEFINE COMMAND REJECTED
```

Defining local dynamic queues

When a local dynamic queue is created using an MQOPEN statement, it is created with SCOPE=SYSTEM. Model 204 sets up control blocks as if you had defined the queue using the DEFINE command.

If the Online is cycled and the permanent local dynamic queue still exists, to access it in a subsequent run, you must issue a DEFINE QUEUE command with the full name of the local dynamic queue, even if the queue manager created the name when the local dynamic queue was originally defined.

Security considerations

Applications creating local dynamic queues via the OPEN QUEUE statement cannot reference model queues unless the system manager defines the model queues.

DEFINE QUEUEMANAGER: Identifying a WebSphere MQ queue manager

Privileges System manager and User 0

Function Identifies an z/OS-WebSphere MQ queue manager that you want to connect to from Model 204. The queue manager must have been previously defined by the z/OS-WebSphere MQ administrator.

Syntax

```
DEFINE {QUEUEMANAGER | QM} name [LIKE previousname]
  WITH SCOPE=SYSTEM
  [{QUEUEMANAGERNAME | QMNAME}=externalqmanname]
```

Where

- *name* identifies the queue manager referred to in the Online. *name* must be:
  - Unique or an error is issued.
  - 1-48 characters long and begin with a letter, followed by letters, numbers, periods, or underscores.
- Use the optional QUEUEMANAGERNAME (or QMNAME) clause if the external name of the queue manager (as defined to WebSphere MQ) differs from the name you refer to in the Online. If you omit QUEUEMANAGERNAME (or QMNAME), the external name of the queue manager is assumed to be *name.*

Usage The external name of the queue manager is validated when a system manager or User 0 issues a START QUEUEMANAGER command. The queue manager has system scope.

You must define a queue manager before you define a queue.

MODIFY QUEUE statement

Function Alters the current default option(s) for an open queue for the issuing user.

Syntax

```
MODIFY QUEUE {%variable | name |
  external_qmanager:external_queue}
  {[option...] [?%variable...]}
```

Where

- Queue to operate on is specified as:

Queue as…	Specifies
%variable	Queue name
name	Literal without quotation marks
external_qmanager:external_queue	External name of a queue manager that contains the remote queue to process and the external name of the remote queue

- *option* is one or more of the following options:
  - ACCEPT_TRUNCATED_MSG or NO_ACCEPT_TRUNCATED_MSG
  - ALL_MSGS_AVAILABLE or NOT_ALL_MSGS_AVAILABLE (default)
  - CONTEXT
  - EXPIRY
  - FAIL_IF_QUIESCING or NO_FAIL_IF_QUIESCING
  - FORMAT
  - LOGICAL_ORDER or NOT_LOGICAL_ORDER (default)
  - MSGTYPE
  - PERSISTENT or NOT_PERSISTENT
  - PERSISTENCE_AS_Q_DEF
  - SYNCPOINT or NO_SYNCPOINT
  - PRIORITY or PRIORITY_AS_Q_DEF
  - REPLY_QMGR
  - REPLY_QUEUE
  - REPORT
  - WAIT or NO_WAIT
  - WAIT_TIME
- *?%variable* specifies option(s) of the MODIFY QUEUE statement to compile at evaluation time. For more information on ?%variables, see “Using runtime options” on page 11.

Usage

Options that apply to MQGET, MQPUT, and MQPUT1 statements can be specified in the DEFINE QUEUE command, and defaults for MQGET and MQPUT statements can be specified in an OPEN QUEUE statement.

You can override defined options by specifying the desired options directly on an MQGET, MQPUT, or MQPUT1 statement.

If you are repeatedly overriding DEFINE QUEUE command options, you might prefer to alter the current options for an open queue using the MODIFY QUEUE statement.

MONITOR MQ: Monitoring MQ/204

Privileges System manager or User 0

Function Aids the system manager in controlling an Online that is using MQ/204. The MQ option provides information on the MQ/204 features.

Syntax

```
MONITOR MQ [(QUEUEMANAGER | QM) qmanager_entity_name
   | (QUEUE | Q) queue_entity_name]
[SUBTASKS] [SUMMARY] [EVERY n]
```

Where

- QUEUEMANAGER (or QM) and QUEUE (or Q) can be used to filter the output of MONITOR MQ:
  - If a QUEUEMANAGER clause is specified, output is restricted to that queue manager.
  - If a QUEUE clause is specified, output is restricted to the specified queue.

  The QUEUEMANAGER, QUEUE, and SUBTASK clauses are mutually exclusive. To monitor the clauses in succession, you must issue a separate MONITOR MQ command for each clause.
- SUBTASKS produces a report of WebSphere MQ subtask activity. The report includes the following numeric output about subtasks:
  - Allocated
  - Currently in use
  - Free, but connected to a queue manger
  - Free and unconnected

  This report also lists the queue managers to which subtasks are currently connected and indicates how many are connected to each.
- SUMMARY limits the output produced by MONITOR MQ:
  - If SUMMARY is specified, the list of users who have each queue open is omitted from the report.
  - If both SUBTASKS and SUMMARY are specified, the list of queue managers and count of subtasks connected to each are omitted from the subtask report.
- EVERY *n* clause provides the command refresh capability.

Usage Output from the MONITOR MQ command goes to USE output, if USE is active.

The MONITOR MQ command lists defined queue managers by both entity and external name, and whether they have been stopped or started. After each queue manager, a list of all queues for that queue manager is displayed, also by entity and external name. After each queue name, a list of the current users is displayed with the following data:

- User number.
- User name.
- Last WebSphere MQ API call made.
- CompCode and Reason display headings for the last call separated by a slash (/) or the word PENDING, if the call has not yet completed. CompCode and Reason values are returned by all WebSphere MQ API calls.
- Number of bytes of data transferred on the last put or get.

Examples The following example shows output from a MONITOR MQ command:

```
> MONITOR MQ
QUEUEMANAGER:  PRODUCTION
EXTERNAL NAME: SYS1.PRODUCTION
STATUS:        STARTED

  QUEUE:          INCOMING
  EXTERNAL NAME: INQUEUE.FOR.WORK

  USER    USERID      LASTCALL  STATUS   DATALEN
  ------  ----------  --------  -------  -------
      25  JOE         MQGET     0/0           50

  QUEUE:          OUTGOING
  EXTERNAL NAME: OUTQUEUE.FOR.WORK

  USER    USERID      LASTCALL  STATUS   DATALEN
  ------  ----------  --------  -------  -------
      25  JOE         MQPUT     0/0           50
      56  FRED        MQPUT1    PENDING       50

QUEUEMANAGER:  TEST
EXTERNAL NAME: SYS1.TEST
STATUS:        STOPPED

  QUEUE:          INBOX
  EXTERNAL NAME: TEST.INBOX

  USER    USERID      LASTCALL  STATUS   DATALEN
  ------  ----------  --------  -------  -------
```

```
      15  SALLY       MQGET     0/0          1000
      23  MIKE        MQGET     2/2033          0

  QUEUE:          OUTBOX
  EXTERNAL NAME: TEST.OUTBOX

QUEUEMANAGER:  TEST2
EXTERNAL NAME: SYS1.TEST2
STATUS:        DEFINED

  QUEUE:          INBOX2
  EXTERNAL NAME: TEST2.INBOX

  QUEUE:          OUTBOX2
  EXTERNAL NAME: TEST2.OUTBOX

>
```

START QUEUEMANAGER: Making queues accessible

Privileges System manager or User 0

Function Validates the existence of a queue manager and marks it available for use.

Syntax `START {QUEUEMANAGER | QM} name`

Where *name* identifies a queue manager created with the DEFINE QUEUEMANAGER command.

Usage If the specified queue manager is already started, the command is ignored.

Once this command has been issued successfully, queues that belong to this queue manager can be opened with the OPEN QUEUE statement described in “OPEN QUEUE statement” on page 82 or MQPUT1 statement described in “MQPUT1 statement” on page 81.

STOP QUEUEMANAGER: Put a queue manager in drain state

Privileges System manager or User 0

Function Places a queue manager in a drain state.

Syntax `STOP {QUEUEMANAGER | QM} name`

Where *name* identifies a queue manager created with a DEFINE QUEUEMANAGER command.

Usage If the queue manager does not exist, or has not been started, an error message is issued. If it has already been stopped, the command is ignored.

Once issued, you cannot open any queues for this queue manager. Attempts to open such queues fail with a $STATUS return code of six (6). However, any current users with open connections to the queue manager are not impacted. You can make the queue manager available again by issuing the START QUEUEMANAGER command.

4

SOUL Statement Reference

SOUL supports MQ/204 as follows:

- OPEN and CLOSE statements support opening and closing queues.
- Set of statements have syntax that closely parallels the WebSphere MQ API for getting and putting messages, and committing and backing out WebSphere MQ transactions.
- MODIFY statement can change defaults on an open queue or alter the size and contents of the data buffer.
- $Functions manipulate queues and queue managers.
- READ IMAGE and WRITE IMAGE statements can manipulate the data buffer.

A SOUL statement with a name that matches a WebSphere MQ API function has a one-to-one correspondence between that SOUL statement and the API call.

To permit an application full flexibility in handling errors, MQ/204 SOUL statements set $STATUS and $STATUSD return codes. See “Error handling with $STATUS and $STATUSD” on page 43.

CLOSE QUEUE statement

Function Closes a queue that was previously opened with an OPEN QUEUE statement.

Syntax

```
CLOSE QUEUE {%variable | M204queue-name
  | external-qmanager:external-queue}
[DELETE | DELETE_PURGE | ?%variable]
```

Where • Queue to operate on is specified as:

Queue as…	Specifies...
%variable	Queue name.
M204queue-name	Queue name as a literal without quotation marks.
external-qmanager:external-queue	External name of a queue manager that contains the remote queue to process and the external name of the remote queue.

- DELETE or DELETE_PURGE and ?%*variable* are the delete options, determined by the attributes assigned in the model queue. A queue with a delete option accepts no more messages or requests and is closed as follows:

Delete option	For local dynamic queue	Deletes the queue…
DELETE	Permanent	When no more messages are on the queue and no uncommitted GET or PUT requests are outstanding.
DELETE_PURGE	Temporary	Purges all outstanding messages and requests.
?%*variable*	Permanent or temporary	At runtime, according to the delete option selected. See “Using runtime options” on page 11.

Usage The CLOSE QUEUE statement performs an MQCLOSE call. If the queue manager is in a drain state, it also performs an MQDISC call.

Deleting local dynamic queues

When a temporary local dynamic queue is closed, the queue is deleted, along with any messages that may still be on it, regardless of the *options* parameter. This is true even if uncommitted MQGET, MQPUT, or MQPUT1 calls, issued using this or another handle, are outstanding against the queue; uncommitted updates that are lost do not cause the unit of work of which they are a part to fail.

After either a temporary or permanent local dynamic queue is deleted, any call (other then a CLOSE QUEUE statement) that attempts to refer to that queue fails with reason code MQRC_Q_DELETED.

When a CLOSE QUEUE statement is issued to delete a permanent local dynamic queue, a security check is made to ensure that the user identifier is authorized to delete the queue.

No check is made when a temporary local dynamic queue is deleted.

Reusing dynamic queue names

After you issue a CLOSE QUEUE DELETE or CLOSE QUEUE DELETE PURGE statement for a queue name, you can reuse the queue name. For example, to reuse the dynamic queue name, CCA1, issue the following statements:

```
CLOSE QUEUE CCA1 DELETE_PURGE
OPEN QUEUE CCA1 DYNAMICQNAME=XNAME MODEL=XMODEL
```

MQ/204 CLOSE statement and the QUEUE keyword

MQ/204 has an upward compatibility issue that is caused by supporting the keyword QUEUE on CLOSE statements. If you have a Model 204 file or group named QUEUE and you issue the SOUL CLOSE statement with it, this might conflict with the CLOSE QUEUE statement, because the FILE and GROUP keywords are optional on SOUL CLOSE statements.

To address this issue, the parsing of the CLOSE statement identifies the case of a file or group named QUEUE. If CLOSE is followed by the keyword QUEUE, the system looks ahead for a token following the word QUEUE:

If…	The statement is assumed to be…
Token is found	MQ/204 CLOSE QUEUE statement.
No token is found after the word QUEUE	CLOSE for a file or group named QUEUE.

Note: If the MQ/204 feature is not linked in, CLOSE statement parsing is unaffected.

MODIFY QUEUE statement

Function Alters the current default option(s) for an open queue for the issuing user.

Syntax

```
MODIFY QUEUE {%variable | name |
  external_qmanager:external_queue}
  {[option...] [?%variable...]}
```

Where

- Queue to operate on is specified as:

Queue as…	Specifies
%variable	Queue name
name	Literal without quotation marks

Queue as…	Specifies
%variable	Queue name
external_qmanager:external_queue	External name of a queue manager that contains the remote queue to process and the external name of the remote queue

- *option* is one or more of the following options:
  - ACCEPT_TRUNCATED_MSG or NO_ACCEPT_TRUNCATED_MSG
  - ALL_MSGS_AVAILABLE or NOT_ALL_MSGS_AVAILABLE (default)
  - CONTEXT
  - EXPIRY
  - FAIL_IF_QUIESCING or NO_FAIL_IF_QUIESCING
  - FORMAT
  - LOGICAL_ORDER or NOT_LOGICAL_ORDER (default)
  - MSGTYPE
  - PERSISTENT or NOT_PERSISTENT
  - PERSISTENCE_AS_Q_DEF
  - SYNCPOINT or NO_SYNCPOINT
  - PRIORITY or PRIORITY_AS_Q_DEF
  - REPLY_QMGR
  - REPLY_QUEUE
  - REPORT
  - WAIT or NO_WAIT
  - WAIT_TIME
- ?%variable specifies option(s) of the MODIFY QUEUE statement to compile at evaluation time. For more information on ?%variables, see “Using runtime options” on page 11.

Usage

Options that apply to MQGET, MQPUT, and MQPUT1 statements can be specified in the DEFINE QUEUE command, and defaults for MQGET and MQPUT statements can be specified in an OPEN QUEUE statement.

You can override defined options by specifying the desired options directly on an MQGET, MQPUT, or MQPUT1 statement.

If you are repeatedly overriding DEFINE QUEUE command options, you might prefer to alter the current options for an open queue using the MODIFY QUEUE statement.

MQBACK statement

Function Backs out pending WebSphere MQ transaction(s), called SYNCPOINT update(s) by WebSphere MQ, on all queue managers or a particular queue manager. This operation is independent of Model 204 file transactions.

Syntax `MQBACK [queuemanentname | %qmvariable]`

Where *queuemanentname* or *%qmvariable* specify to back out a particular queue manager.

Usage The MQBACK statement operates on active WebSphere MQ transactions for the issuing user only. Active WebSphere MQ transactions for other users are not affected.

An MQBACK statement without a queue manager specified backs out all active transactions for all queue managers, processing in alphabetical order by name. You can back out a particular queue manager by specifying its name as a literal without surrounding quotation marks or in a %variable.

If a failure occurs when processing multiple queue managers, the operation stops at the queue manager that had the error. You can use the $MQ_LASTQUEUEMANAGER_ENTITY function to determine which queue manager had the error.

MQCMIT statement

Function Commits pending WebSphere MQ transactions, called SYNCPOINT update(s) by WebSphere MQ, on all queue managers or a particular queue manager. This operation is independent of Model 204 file transactions.

Syntax `MQCMIT [queuemanentname | %qmvariable]`

Where *queuemanentname* or *%qmvariable* specify to commit all updates to a particular queue manager.

Usage The MQCMIT statement operates on active WebSphere MQ transactions for the issuing user only. Active WebSphere MQ transactions for other users are not affected.

You can limit the commit to a particular queue manager by specifying its name as a literal without surrounding quotation marks or in a %variable.

MQCMIT statement without a queue name specified commits all active transactions for all queue managers, processing in alphabetical order by name.

If a failure occurs when processing multiple queue managers, the operation stops at the queue manager that had the error. You can use the

$MQ_LASTQUEUEMANAGER_ENTITY function to determine which queue manager had the error.

MQGET statement

Function Retrieves a message from a currently open queue.

Syntax

```
MQGET {(image[,image]...) | %variable | BUFFER
| MQ_BUFFER}
[RFH2=(image | BUFFER)]
[FROM] {%qvariable | entname
| external_queuemanager:external_queue}
[BUFLEN={%bvar | n}]
[MSGLEN=%mvar]
[[option...] [?%variable...]]
```

Where

- *image* or *%variable* or BUFFER (formerly and still accepted MQ_BUFFER) specifies the target into which the message data is placed. You can specify up to 10 images.

If the target is…	Then…
image	Image item READLEN is filled in with the number of bytes of message data read into the image
%variable	Number of bytes read can be obtained with $LEN
BUFFER (formerly and still accepted MQ_BUFFER)	Message is simply left in the Universal Buffer, where is remains intact until the next MQ/204 statement is issued
RFH2=(image | BUFFER)	Java RFH2 header can be accommodated.

- FROM clause identifies the queue to operate on as specified:

Queue as…	Specifies…
%qvariable	Queue name
entname	Literal without quotation marks
external_qmanager:external_queue	External name of a queue manager that contains the remote queue to process and the external name of the remote queue

- BUFLEN can determine the destination size of BUFFER area in bytes, as follows:
  - If BUFLEN was specified and the size of your BUFFER area is less than BUFLEN, BUFFER area is resized to the value of BUFLEN. Otherwise BUFFER size is unchanged.
  - If BUFLEN was not specified:

Destination byte size for…	**Is…**
String %variable	Declared length of the %variable.
Image	Maximum length of the image.
Multiple images	Sum of the individual image sizes.

    If the size of your BUFFER area is less than the destination size, the existing BUFFER area is deleted and a new BUFFER area allocated, with a size equal to the destination size.

- *option* is one or more of the following options; see Table 5-1 on page 87 for a full discussion of each option:
  - ACCEPT_TRUNCATED_MSG or NO_ACCEPT_TRUNCATED_MSG
  - ALL_MSGS_AVAILABLE_ or NOT_ALL_MSGS_AVAILABLE (default)
  - BROWSE_FIRST
  - BROWSE_MSG_UNDER_CURSOR
  - BROWSE_NEXT
  - CONVERT or NO_CONVERT
  - CORRELID=value
  - DESCRIPTOR=image
  - FAIL_IF_QUIESCING or NO_FAIL_IF_QUIESCING
  - GROUPID=%variable
  - GSTATUS=%variable
  - LOCK or UNLOCK
  - LOGICAL_ORDER or NOT_LOGICAL_ORDER (default)
  - MATCH_CORREL_ID (default) or NOT_MATCH_CORREL_ID
  - MATCH_GROUP_ID
  - MATCH_MSG_ID (default) or NOT_MATCH_MSG_ID
  - MATCH_MSG_SEQ_NUMBER
  - MATCH_MSG_TOKEN
  - MARK_SKIP_BACKOUT
  - MSGID=value
  - MSGTOKEN=%variable

- MSG_UNDER_CURSOR
- NEW_CORREL_ID
- SEQUENCE=%variable
- SYNCPOINT or NO_SYNCPOINT or SYNCPOINT_IF_PERSISTENT
- WAIT or NO_WAIT
- WAIT_TIME=value

- ?%*variable* specifies *option(s)* of the MQGET statement to compile at evaluation time. For more discussion of ?%variables, see “Using runtime options” on page 11.

Analyzing an MQGET statement

When you issue an MQ/204 MQGET statement, the WebSphere MQ MQGET is issued. WebSphere MQ places a message or part of a message in your BUFFER area and returns:

- Actual length of the message that was retrieved or partially retrieved
- Completion code and reason code that qualifies the completion code

- The length of the message is recorded as follows:

Argument or function	Is set to…
$BUFFER_USED	Length of the data retrieved into BUFFER area
$MQ_MESSAGE_LEN	Actual length of the message
MSGLEN=%mvar	Actual length of the message

- If the completion code indicates successful completion:

If the destination is…	Then…
BUFFER area	$BUFFER_POSITION is set to 1.
Anything other than BUFFER area	Message is copied from BUFFER area to the destination.
Image or images	Each READLEN is set.

$BUFFER_POSITION is set to the byte after the last byte copied.

- If the completion code indicates unsuccessful completion:
  - $STATUS is set appropriately
  - $STATUSD is set to the WebSphere MQ reason code
  - $BUFFER_POSITION is set to 1
  - No further processing is done

Using the BUFFER area

Data transfers between WebSphere MQ and Model 204 are performed via the BUFFER area. You can perform all MQ/204 operations without directly referencing the BUFFER area. However, for maximum flexibility, you can directly manipulate the contents of the BUFFER area using the following statement forms:

- MQGET BUFFER
- MQPUT BUFFER
- READ IMAGE FROM BUFFER
- WRITE IMAGE ON BUFFER

Each user thread of MQ/204 has only one BUFFER area, which is private to that thread. BUFFER area resides in a separately allocated area of memory; it does not occupy space in the user's server, nor in CCATEMP. BUFFER area is allocated automatically for you when required, and deallocated when you log off.

When an MQGET statement retrieves a message from WebSphere MQ to a %variable, image, or image list, the data is first moved from WebSphere MQ to BUFFER area, and then copied to the %variable, image, or image list. When an MQPUT or MQPUT1 statement sends a message from a %variable, image, or image list to WebSphere MQ, the data is first copied from the %variable, image, or image list to BUFFER area and then moved to WebSphere MQ.

Handling the end of a request

- Under the usual circumstances, at request end, the user's BUFFER area remains allocated but its contents are cleared.
- In an APSY subsystem with AUTOCOMMIT=NO, the user’s BUFFER area remains allocated and its contents are unchanged.

Usage Options in common with OPEN QUEUE and DEFINE QUEUE statements inherit default values from the OPEN QUEUE statement, which in turn inherits from the queue:

- If DESCRIPTOR=*image* is specified, the MQMD message descriptor is returned, even if the MQGET failed. However, if the MQGET failed, the contents of *image* may or may not be meaningful, depending on the nature of the failure.
- After a get operation with BUFFER, the data can be loaded into an image with an extension to the READ IMAGE statement. See “Universal Buffer statements” on page 85.

- When a local dynamic queue is deleted, any MQGET statements with the WAIT option that are outstanding against the queue are canceled and reason code MQRC_Q_DELETED is returned.

Using the browse options

On queues open for browsing only, if neither BROWSE_FIRST nor BROWSE_NEXT is specified, then the default operation of MQGET is BROWSE_FIRST on the first MQGET after the OPEN QUEUE and BROWSE_NEXT on all subsequent MQGET statements.

On queues open for both browsing and input, if neither BROWSE_FIRST nor BROWSE_NEXT is specified, then, as the MQGET statement retrieves messages, they are deleted from the queue. This is consistent with the behavior of the WebSphere MQ API.

- On get operations, the READLEN item of each image is set. If the data runs out before filling all the images, then READLEN is set to zero for the remaining images.
- If BUFLEN is specified, BUFLEN is used to limit the number of bytes retrieved from WebSphere MQ.
- MSGLEN is an output %variable; its value before the MQGET is irrelevant.

Truncated messages

Truncation of messages can occur at two different points:

- When the data is moved from WebSphere MQ to the BUFFER area ($STATUS=12):

```
($MQ_MESSAGE_LEN > $BUFFER_USED)
```

- When the data is copied from the BUFFER area to the destination ($STATUS=14):

```
($MQ_MESSAGE_LEN = $BUFFER_USED)
```

If truncation occurs when the data is moved from WebSphere MQ to the BUFFER area, the setting of the ACCEPT_TRUNCATED_MSG option determines whether or not WebSphere MQ leaves the message on the queue; or, if in browse mode, whether or not WebSphere MQ advances the browse cursor.

Error handling consideration

MQGET statement processing always retrieves the message from WebSphere MQ into the user's BUFFER area.

- If the WebSphere MQ call fails, $STATUS is set to 12, $STATUSD is set to the WebSphere MQ reason code, and the operation ends.

- If the call succeeds, the message is copied from the BUFFER area into the target—%variable, image, or image list.
- If the target is not large enough to hold the message retrieved, $STATUS is set to 14 to indicate truncation; otherwise, $STATUS is set to 0 to indicate success.

Under some circumstances, the WebSphere MQ call can fail but still return a message or part of a message. In those cases, $STATUS is set to 12, $STATUSD is set to the WebSphere MQ reason code, but the operation continues and the message is copied from the BUFFER area into the target.

If the target is not big enough to hold the message (or partial message) retrieved, $STATUS is not reset to 14, but instead retains its original value. In other words, $STATUS and $STATUSD always report the first error, and the subsequent error is not explicitly reported.

The circumstances under which a WebSphere MQ MQGET statement can fail, but still return a message or part of a message are:

WebSphere MQ reason code	Message
2079	TRUNCATED_MSG_ACCEPTED
2080	TRUNCATED_MSG_FAILED
2110–2120	Various conversion errors

If an MQGET statement returns $STATUS=12 and $STATUSD=2079, or 2080, or 2110–2120, and the target is a %variable, image or image list, then the program *must* determine whether the message was truncated when copied to the target. You cannot use $STATUS=14.

You can use the following after an initial error to determine if truncation also occurred.

```
  IF ($BUFFER_POSITION LE $BUFFER_USED) THEN
* We truncated moving from the buffer to the %variable
* or image target
  ELSE
* We have $BUFFER_POSITION = $BUFFER_USED + 1
* We didn't truncate moving the buffer to the %variable
* or image target
  END IF
```

Note: In the case of $STATUSD=2079 or 2080, the message was truncated when moved from WebSphere MQ to the BUFFER area, but could not have been truncated again when copied from the BUFFER area to the target.

Handling an incoming message with an RFH2 header

If you expect the incoming message to contain an RFH2 header, use the RFH2=(*image* | BUFFER) option. You can specify an image into which the header will be copied, or you can specify BUFFER, in which case the RFH2 header is left in the BUFFER, preceding the data.

The RFH2 header, if the target is an image, is copied from the buffer to an image, but the RFH2 header is not removed from the buffer—it still precedes the data.

- If the target for the RFH2 header specified BUFFER, the RFH2 header is simply left at the beginning of the buffer.
- If the target of the data is specified as an image or %variable, the data is copied from its position behind the RFH2 header (if present) to the image or %variable.
- If the target of the data is specified BUFFER, it is left in place in the buffer behind the RFH2 header.

In all cases where the RFH2 header is involved, the buffer after the operation contains the RFH2 header followed by the data, and $BUFFER_POSITION variable has the position (=offset+1) of the data, if the data is left in the buffer, or beyond the data if the data has been copied to an image, image list or variable.

The $BUFFER_USED has the total length of RFH2 header and data.

Using RFH2 keyword with MQGET

In all cases of GET, MSGLEN is set to the length of the data received, excluding the length of the RFH2 header if any.

GET target	RFH2 in image	RFH2 in buffer
Data in image or %variable	$BUFFER_USED set to length of RFH2 header, plus length of data. $MQ_MESSAGE_LEN set to length of data. $BUFFER_POSITION set to byte beyond last byte of data copied. RFH2 copied to image. Data copied to image or %variable.	$BUFFER_USED set to length of RFH2 header, plus length of data. $MQ_MESSAGE_LEN set to length of data.$BUFFER_POSITION set to byte beyond last byte of data copied. Data copied to image or %variable.

GET target	RFH2 in image	RFH2 in buffer
Data in buffer	$BUFFER_USED set to length of RFH2 header + length of data. $MQ_MESSAGE_LEN set to length of data. $BUFFER_POSITION set to first byte of data (the byte beyond the RFH2 header). RFH2 copied to image.	$BUFFER_USED set to length of RFH2 header + length of data. $MQ_MESSAGE_LEN set to length of data. $BUFFER_POSITION set to first byte of data (the byte beyond the RFH2 header).

MQPUT statement

Function Places a message on a currently open queue.

Syntax

```
MQPUT {(image[,image1]...) | %variable | 'string'
  | BUFFER | MQ_BUFFER}
[RFH2=(image | BUFFER)]
[ON] {%qvariable | entname
| external_qmanager:external_queue}
[BUFLEN= {%bvar | n}]
[[option...] [?%variable...]]
```

Where

- *image* or *%variable* or *BUFFER* (formerly and still accepted MQ_BUFFER) specifies the target into which the message data is placed. You can specify up to 10 images.

If the target is…	Then…
Image	To swap images in and process serially. You can specify up to 10 images.
%variable	Current value of variable.
'string'	Literal with quotation marks.
BUFFER (formerly, and still accepted as MQ_BUFFER)	Message is preloaded in the Universal Buffer area.
RFH2=(image | BUFFER)	Java RFH2 header can be accommodated.

- Queue to operate on is:

Queue as…	Specifies...
%qvariable	Queue name.
entname	Literal without quotation marks.
external_qmanager:external_queue	External name of a queue manager that contains the remote queue to process and the external name of the remote queue.

- BUFLEN specifies in bytes, %bvar or n, the size of BUFFER area.
- *option* is one or more of the following options:
  - ACCOUNTINGTOKEN
  - APPLIDENTITYDATA=value
  - APPLORIGINDATA=value
  - CONTEXT=value or NO_CONTEXT
  - CORRELID=value
  - DESCRIPTOR=image
  - DEFAULT_CONTEXT
  - EXPIRY=value
  - FAIL_IF_QUIESCING or NO_FAIL_IF_QUIESCING
  - FEEDBACK=value
  - FORMAT=value
  - GROUPID=%variable
  - GSTATUS=%variable
  - LOGICAL_ORDER or NOT_LOGICAL_ORDER (default)
  - MSGID=value
  - MSGTYPE=value
  - PASS_ALL_CONTEXT
  - PASS_IDENTITY_CONTEXT
  - PERSISTENT or NOT_PERSISTENT
  - PERSISTENCE_AS_Q_DEF
  - PRIORITY=value or PRIORITY_AS_Q_DEF
  - PUTAPPLNAME=value
  - PUTAPPLTYPE=value
  - PUTDATE=value
  - PUTTIME=value

- REPLY_QMGR=name
- REPLY_QUEUE=name
- REPORT=options
- SEQUENCE=%variable
- SET_ALL_CONTEXT
- SET_IDENTITY_CONTEXT
- SYNCPOINT or NO_SYNCPOINT
- USERIDENTIFIER='value'

?%variable specifies option(s) of the MQPUT and MQPUT1 statements to compile at evaluation time.

Usage If an MQPUT or MQPUT1 statement sends a message from a %variable, image, or image list to WebSphere MQ, the data is first copied from the %variable, image, or image list to the BUFFER area, and then copied to WebSphere MQ.

MQPUT and MQPUT1 processing

When WebSphere MQ MQPUT or MQPUT1 is issued, WebSphere MQ:

- Copies the message from the user's BUFFER area
- Returns a completion code and reason code (qualifies the completion code)

Irrespective of the completion code:

- $MQ_MESSAGE_LEN is set to the length of the message
- $BUFFER_POSITION is set to 1

If the completion code does not indicate successful completion, $STATUS is set appropriately, and $STATUSD is set to the WebSphere MQ reason code.

Usage notes for options

- Options in common with the DEFINE QUEUE command and the OPEN QUEUE statement inherit default values from the OPEN QUEUE statement, which in turn inherits from the queue.
- You can specify the following identity context options only if you also specify the SET_ALL_CONTEXT or SET_IDENTITY_CONTEXT option:
  - ACCOUNTINGTOKEN
  - APPLIDENTITYDATA
  - USERIDENTIFIER
- You can specify the following origin context parameters only if you also specify the SET_ALL_CONTEXT option:

  - APPLORIGINDATA
  - PUTAPPLNAME
  - PUTAPPLTYPE
  - PUTDATE
  - PUTTIME

- If DESCRIPTOR=*image* is specified, the MQMD message descriptor is returned, even if the MQPUT or MQPUT1 failed. However, if the MQPUT or MQPUT1 failed, the contents of *image* might or might not be meaningful, depending on the nature of the failure.
- On each MQPUT statement, the message context is set by specifying one of the following, mutually exclusive options.
  - DEFAULT_CONTEXT
  - NO_CONTEXT
  - PASS_ALL_CONTEXT
  - PASS_IDENTITY_CONTEXT
  - SET_ALL_CONTEXT
  - SET_IDENTITY_CONTEXT

  If none of the previous options is specified, WebSphere MQ applies its own default, which is DEFAULT_CONTEXT. For more information on message context see “Applying date and time-stamps to messages” on page 79.

Managing BUFFER area

If the message source is BUFFER area:

If…	Then…
$BUFFER_USED=0	Operation returns $STATUS=26
BUFLEN=%*bvar* was not specified	Message length is $BUFFER_USED
BUFLEN=%*bvar* was specified	Message length is the minimum of $BUFFER_USED and %*lvar*

If the source is anything other than BUFFER area, the source size is determined:

For	Size is…
String %variable	Current length of the %variable
Image	Maximum length of the image
Multiple images	Sum of the individual image sizes

If the size of the user's BUFFER area is less than the message length, the existing BUFFER area is deleted and a new BUFFER area allocated, with a size equal to the message length.

The message is copied from the source to the BUFFER area.

If BUFLEN=%bvar was…	Message length is…
Not specified	Source size
Specified	Minimum of the source size and %*lvar*

$BUFFER_USED is set to the message length.

Applying date and time-stamps to messages

Each WebSphere MQ message is associated with two origin context fields called PUTDATE and PUTTIME, which act as a date-stamp and time-stamp, respectively, when a message is posted to a queue.

When a message is put on a queue, these fields are usually filled in. Any WebSphere MQ-enabled program issuing a WebSphere MQ MQPUT statement can explicitly set values for these fields, or alternatively (and most commonly), the values can be set by the WebSphere MQ queue manager. In MQ/204, you explicitly set values for these fields by specifying the option PUTDATE and/or PUTTIME on the MQPUT and MQPUT1 SOUL statements. The queue manager sets values for those fields, if you do no specify those options.

When a message is received from a queue, the values of these fields that were set by the message originator are passed in the WebSphere MQ message descriptor, MQMD. In MQ/204, you can obtain a copy of the message descriptor by specifying option DESCRIPTOR=*imagname* on the MQGET SOUL statement. The MQMD is not available to you, if you do not specify this option.

If you decide to set or inspect the values of the PUTDATE and PUTTIME fields in your own application, read the IBM manual *WebSphere MQ Application Programming Reference* to understand the format of these fields when their values are set by a WebSphere MQ queue manager. In particular, according to this manual:

> "Greenwich Mean Time (GMT) is used for the PutDate and PutTime fields, subject to the system clock being set accurately to GMT. On OS/2, the queue manager uses the TZ environment variable to calculate GMT."

If you set PUTDATE and PUTTIME in your own applications, WebSphere MQ lets you use any date or time format, any time zone, or even invalid values. Rocket strongly recommends, however, that you adhere to the default format used by WebSphere MQ.

Handling an outgoing message with an RFH2 header

Use the RFH2 option if you want the outgoing message to include an RFH2 header.

You can specify an image from which the header will be copied, or you can specify BUFFER, in which case the RFH2 header must be at the start of the BUFFER (position = 1), ahead of the data if that is also sourced from the BUFFER.

- If the RFH2 keyword is present, the FORMAT field will be set to MQHRF2.
- If the RFH2 keyword is present, but there is no valid RFH header in the specified image or buffer, the statement will fail with $STATUS = 47.

The RFH2 header, if specified, is taken from an image and placed in the buffer, or, if BUFFER is specified, is presumed to be at position 1 in the buffer (this will be checked).

- The data, if sourced from an image or %variable, is copied to the buffer so that it immediately follows the RFH2.
- If the data is sourced from the buffer, it is presumed to follow the RFH2 (if the RFH2 header is also sourced from the buffer), or is copied so that it follows the RFH2 header (if the RFH2 header is sourced from an image).

In all cases where the RFH2 header is involved, the buffer after the operation contains the RFH2 header followed by the data.

Using RFH2 keyword with MQPUT

PUT source	RFH2 from image	RFH2 from buffer
Data from image or %variable	The RFH2 is copied to the buffer. Then the data is appended to the RFH2 header in the buffer.	The beginning of the buffer is checked for a valid RFH2 header. The data is appended to the RFH2 header in the buffer.
Data from buffer	The data is copied downwards in the buffer to make way for the RFH2 header, which is then copied to the beginning of the buffer.	The RFH2 is assumed to precede the data in the buffer. The beginning of the buffer is checked for a valid RFH2 header.

In all cases of PUT:

- $BUFFER_POSITION is ignored, and set to 1 after the operation.
- $BUFFER_USED is set to the length of RFH2 header, plus the length of data.
- $MQ_MESSAGE_LEN set to length of data.
- MSGLEN set to length of data.

- $BUFFER_POSITION set to first byte of data beyond the RFH2 header.

MQPUT1 statement

Function Places one message on a queue that is not currently open. The MQPUT1 statement is a combination of OPEN QUEUE, MQPUT, and CLOSE QUEUE. It is the most efficient way to put a single message on a queue.

Syntax

```
MQPUT1 {(image [,image1]…) | %variable | 'string'
  | BUFFER | MQ_BUFFER}
[ON] {%qvariable | entname
  | external_qmanager:external_queue} [option …]
```

Where

- *image*, *%variable*, *'string'* , or *BUFFER* (formerly, and still accepted, MQ_BUFFER) area specified contains the message data. (See “Queue security processing” on page 15.)

This message source…	Specifies…
image	To swap images in and process serially. You can specify up to 10 images.
%variable	Current value of variable.
'String'	Literal with quotation marks.
BUFFER (or MQ_BUFFER) area	Message is preloaded in the Universal Buffer area.

- Queue to operate on is specified as:

Queue as…	Specifies...
%qvariable	Queue name.
entname	Literal without quotation marks.
external_qmanager:external_queue	External name of a queue manager that contains the remote queue to process and the external name of the remote queue.

- *option* is one or more of the options.

  Options are the same as those for MQPUT with one addition: the PASS_USER_CONTEXT option. See the list in “MQPUT statement” on page 75.

  Also, LOGICAL_ORDER is not a permitted option for MQPUT1.

MQPUT1 inherits options directly from the queue, because it does its own open and is completely independent from the OPEN QUEUE statement.

Usage

- The security considerations discussed in “Queue security processing” on page 15 apply to MQPUT1 as well.
- If the queue named in an MQPUT1 statement is already open, the statement is processed regardless; there is no error.
- On each MQPUT statement, the message context is set by specifying one of the following, mutually exclusive options.
  - DEFAULT_CONTEXT
  - NO_CONTEXT
  - PASS_ALL_CONTEXT
  - PASS_IDENTITY_CONTEXT
  - SET_ALL_CONTEXT
  - SET_IDENTITY_CONTEXT

  If none of the previous options is specified, WebSphere MQ applies its own default, which is DEFAULT_CONTEXT. For more information on message context see “Applying date and time-stamps to messages” on page 79.

OPEN QUEUE statement

Function Opens a previously defined queue.

Syntax

```
OPEN QUEUE {%variable | entname |
  external_qmanager:external_queue}
[[DYNAMICQNAME=%variable | literal]
[MODEL=%variable | literal]]
[[option...] [?%variable...]]
```

Where

- Queue to operate on is specified as:

Queue as…	Specifies...
%variable	Queue name.
entname	Literal without quotation marks.

Queue as…	Specifies...
external_qmanager:external_queue	External name of a queue manager that contains the remote queue to process and the external name of the remote queue.

- DYNAMICQNAME keyword value is the internal Model 204 name from a DEFINE QUEUE command of a preallocated WebSphere MQ model queue that was defined with the WebSphere MQ MQQDT_PERMANENT_DYNAMIC or MQQDT_TEMPORARY_DYNAMIC attribute. See “Local dynamic queue support” on page 8.

  You can specify a DYNAMICQNAME value in one of the following ways:

  – Give the fully qualified name. It will be placed in the DYNAMICQNAME field of the WebSphere MQ MQOD structure.

  – Specify a prefix of less than 33 characters for the name, followed by an asterisk (*). The queue manager generates the rest of the name.

    For example, you might want each user to use a certain prefix, or you might want to give a special security classification to queues with a certain prefix in their name.

  – Let the queue manager generate the full name. To use this method, specify an asterisk (*) in the first character position of the DYNAMIC-QNAME field.

  You must specify the DYNAMICQNAME and MODEL keywords together, otherwise MQ/204 issues the following message:

  ```
  M204.2516: MQ/204 OPTION ERROR: options REQUIRE options
  ```

- MODEL keyword value is the internal, Model 204 name of the model queue.

  When you specify the MODEL keyword you must also specify the DYNAMICQNAME keyword, otherwise MQ/204 issues the following message:

  ```
  M204.2516: MQ/204 OPTION ERROR: options REQUIRE options
  ```

- *option* is one or more of the following options; see Table 5-1 on page 87 for a full discussion of each option:

  – ACCEPT_TRUNCATED_MSG or NO_ACCEPT_TRUNCATED_MSG

  – ALL_MSGS_AVAILABLE or NOT_ALL_MSGS_AVAILABLE (default)

  – ALLOW_PASS_ALL_CONTEXT

  – ALLOW _PASS_IDENTITY_CONTEXT

  – ALLOW_SET_ALL_CONTEXT

  – ALLOW_SET_IDENTITY_ONTEXT

  – BROWSE

  - CONTEXT
  - EXPIRY
  - FAIL_IF_QUIESCING or NO_FAIL_IF_QUIESCING
  - FORMAT
  - INPUT_AS_Q_DEF (default for local queues) or INPUT_EXCLUSIVE or INPUT_SHARED
  - LOGICAL_ORDER or NOT_LOGICAL_ORDER (default)
  - MSGTYPE
  - OUTPUT (default for remote queues)
  - PASS_USER_CONTEXT
  - PERSISTENT or NOT_PERSISTENT or PERSISTENCE_AS_Q_DEF
  - PRIORITY or PRIORITY_AS_Q_DEF
  - REPLY_QMGR
  - REPLY_QUEUE
  - REPORT
  - SAVE_ALL_CONTEXT
  - WAIT or NO_WAIT
  - WAIT_TIME

- ?%*variable* specifies *option(s)* of the OPEN QUEUE statement to compile at evaluation time. For more discussion of ?%variables, see “Using runtime options” on page 11.

Usage The queue manager for the queue must already be started. If the queue is the first queue that the user is opening for a queue manager, then MQ/204 attempts to connect the user to the queue manager prior to opening the queue. This statement performs a WebSphere MQ MQOPEN operation, and manages connections to the queue managers as needed using MQCONN and MQDISC.

Simultaneous input and output are supported, as are browse and output.

MQ/204 OPEN statement and QUEUE keyword

MQ/204 has an upward compatibility issue that is caused by supporting the keyword QUEUE on OPEN statements. If you have a Model 204 file or group named QUEUE and you use the SOUL OPEN statement with it, this might conflict with the OPEN QUEUE statements, because the FILE and GROUP keywords are optional on OPEN statements.

To address this issue, the parsing of OPEN statements identifies the case of a file or group named QUEUE. If OPEN is followed by the keyword QUEUE, the system looks ahead for a token following the word QUEUE:

If…	The statement is assumed to be…
Token is found	MQ/204 OPEN QUEUE statement.
No token is found after the word QUEUE	OPEN for a file or group named QUEUE.

Note: If the MQ/204 feature is not linked in, OPEN statement parsing is unaffected.

Specifying a local dynamic queue name

The name of the model queue in the WebSphere MQ ObjDesc parameter is replaced with the name of the local dynamic queue created when the call has completed.

When a local dynamic queue is created via MQOPEN, Model 204 calls the routines to set up control blocks as if the user had entered the DEFINE QUEUE *dynamic-queue* command and is updated with the fully resolved name in the instances where the local dynamic queue name ended with an asterisk.

When you issue an OPEN QUEUE statement that did not create the local dynamic queue by referencing a model queue, a partially qualified name prefix cannot compile.

Opening a remote queue

You can open a remote queue with only the remote queue default option, OUTPUT, set. All other options are invalid.

Universal Buffer statements

The following statements manage the size and contents of the Universal Buffer:

- The MODIFY BUFFER statement, formerly MODIFY MQ_BUFFER, manages the size of the Universal Buffer and also keeps messages or overwrites them with fill characters.
- The READ IMAGE statement reads data until the Universal Buffer is exhausted.
- The WRITE IMAGE statement loads data into the Universal Buffer prior to a PUT operation that specifies BUFFER.

These statements are documented in the Model 204 documentation wiki (http://m204wiki.rocketsoftware.com/index.php/Images).

5

MQ/204 Options for Commands and Statements

The definition and activity of MQ/204 queue managers, queues, and messages can be defined and changed using options. For review of how options are changed and passed by MQ/204 statements and commands, see “Rules of inheritance” on page 9.

MQ/204 options

The options in Table 5-1 apply to the commands and statements listed.

Note: You cannot use *option=%variable* in a DEFINE QUEUE command.

Table 5-1. Options for MQ/204 commands and statements

Option	Purpose	Discussion	Can be set on…
ACCEPT_ TRUNCATED_MSG	If a message was truncated because the target was too small, removes it from the queue; or, if browsing, advances the browse cursor.	Corresponds to the WebSphere MQ option MQGMO_ACCEPT_ TRUNCATED_MSG.	DEFINE QUEUE MODIFY QUEUE MQGET OPEN QUEUE
NO_ACCEPT_ TRUNCATED_MSG (default)	Truncated message is not removed from the queue; the browse cursor is not advanced.	This option is the alternate to ACCEPT_TRUNCATED_ MSG option.	DEFINE QUEUE MODIFY QUEUE MQGET OPEN QUEUE
ACCOUNTINGTOKEN =*’value’* or =%*variable*	Sets the value of the Accounting Token portion of the identity context.	String from 1 to 32 characters long.	MQPUT MQPUT1

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
ALL_MSGS_ AVAILABLE	Use on an MQGET call to retrieve only messages that belong to complete groups, as well as messages that do not belong to a group.		DEFINE QUEUE MODIFY QUEUE MQGET OPEN QUEUE
NOT_ALL_MSGS_AV AILABLE (default)	Use on an MQGET call to retrieve any message in the queue, including partial groups.		DEFINE QUEUE MODIFY QUEUE MQGET OPEN QUEUE
ALLOW_PASS_ALL_ CONTEXT	Passes origin context information on all subsequent MQPUT calls.	Corresponds to WebSphere MQ MQOO_PASS_ALL_ CONTEXT. Can be specified only if the OUTPUT option is also specified.	OPEN QUEUE
ALLOW _PASS_ IDENTITY_ CONTEXT	Passes identity context information on all subsequent MQPUT calls.	Corresponds to WebSphere MQ MQOO_PASS_IDENTITY _ CONTEXT. Can be specified only if the OUTPUT option is also specified.	OPEN QUEUE
ALLOW_SET_ALL_ CONTEXT	Sets origin context information on all subsequent MQPUT calls.	Corresponds to WebSphere MQ MQOO_SET_IDENTITY_ CONTEXT. Can be specified only if the OUTPUT option is also specified.	OPEN QUEUE
ALLOW_SET_ IDENTITY_ CONTEXT	Sets identify context information on all subsequent MQPUT calls.	Corresponds to WebSphere MQ MQOO_SET_IDENTITY_ CONTEXT. Can be specified only when the OUTPUT option is also specified.	OPEN QUEUE
APPLIDENTITYDATA =*'value'* or =%*variable*	Sets the value of the ApplIdentityData portion of the identity context.	String from 1 to 32 characters long. Defaults to blanks.	MQPUT MQPUT1

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
APPLORIGINDATA *='value'* or =%*variable*	Sets the value of the ApplOriginData portion of the origin context.	String from 1 to 4 characters long. Defaults to blanks.	MQPUT MQPUT1
BROWSE	Examines messages without removing them from the queue.	Corresponds to WebSphere MQ MQOO_BROWSE. Compatible with all other access options.	OPEN QUEUE
BROWSE_FIRST	First message is retrieved, but not deleted from the queue: • If either MSGID or CORRELID is specified, the first matching message on the queue is browsed. • If neither MSGID nor CORRELID is specified, then the first message on the queue is browsed.	Corresponds to WebSphere MQ MQGMO_BROWSE_FIRST; is mutually exclusive with BROWSE_NEXT. You can specify this option only if the queue was opened in browse mode.	MQGET
BROWSE_MSG_ UNDER_CURSOR	Reread the same message again.	You would typically do this to unlock a locked message or vice versa. See the LOCK and UNLOCK options later in this table.	MQGET
BROWSE_NEXT	Next message, relative to the current browse position, is retrieved, but not deleted from the queue: • If either MSGID nor CORRELID is specified, the next matching message on the queue is browsed. • If neither MSGID or CORRELID is specified, then the next message on the queue is browsed.	Corresponds to WebSphere MQ MQGMO_BROWSE_NEXT; is mutually exclusive with BROWSE_FIRST. You can specify this option only if the queue was opened in browse mode.	MQGET
BUFLEN=*n* or =*%variable*	Specifies the maximum number of bytes to get or put.	Resized according to size of BUFFER (formerly and still accepted MQ_BUFFER) area.	MQGET MQPUT MQPUT1

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
CONTEXT=*name* or =*%variable*	Passes name of a queue entity from which context information should be passed.	Context can be passed only from a queue that is opened for input with the SAVE_ALL_CONTEXT option.	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
CONVERT (default)	Converts messages from one system to another for the convenience of the recipient; from EBCDIC to ASCII, for example.	Corresponds to WebSphere MQ MQGMO_CONVERT.	MQGET
CORRELID='*msgid* ' *or* =%*variable*	Value to place in the correlation ID field of the message. On a reply message, the ID of the request being replied to (MQMD.MsgId) is often set in this field. However, the correlation ID might be set on any message type.	String from 1 to 24 characters long containing a correlation ID to use as matching criterion. Default is blanks. Corresponds to WebSphere MQ MQMD.CorrelId. You can specify the value as a %variable.	MQGET MQPUT MQPUT1
DEFAULT_CONTEXT	Message is put with the default context information.	Corresponds to WebSphere MQ option MQPMO_DEFAULT_ CONTEXT. Mutually exclusive with: • NO_CONTEXT • PASS_ALL_CONTEXT • PASS_IDENTITY_ CONTEXT • SET_ALL_CONTEXT • SET_IDENTITY_ CONTEXT	MQPUT MQPUT1
DELETE	Deletes a permanent local dynamic queue when there are no more messages on the queue and no uncommitted MQGET or MQPUT requests outstanding.	Corresponds to WebSphere MQ option MQCO_DELETE.	CLOSE QUEUE
DELETE_PURGE	Deletes a temporary local dynamic queue and purges all outstanding messages and requests.	Corresponds to WebSphere MQ option MQCO_DELETE_PURGE.	CLOSE QUEUE

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
DESCRIPTOR=*image* (optional)	Image in which to place the message descriptor (MQMD structure). (See “MQPUT statement” on page 75.)	A given application might not need the message descriptor. If the size of this image is less than the size of an MQMD, then the MQMD is truncated.	MQGET MQPUT MQPUT1
DYNAMICQNAME =*name* or =%*variable*	Specifies the name of a preallocated WebSphere MQ model queue.	Must be set with the MODEL option.	MQOPEN
EXPIRY=*n* *or* =UNLIMITED *or* =%*variable*	Expiration period of the message in tenths of a second.	Corresponds to MQMD.Expiry field. The default is UNLIMITED (no expiration).	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
FAIL_IF_QUIESCING	Operation fails if the queue manager is inactive.	Corresponds to the WebSphere MQ MQGMO_FAIL_IF_ QUIESCING option.	DEFINE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
NO_FAIL_IF_ QUIESCING (default)	Operation waits if the queue manager is inactive.	This option is the alternate to FAIL_IF_QUIESCING option.	DEFINE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
FEEDBACK=NONE (default) *or* =*n* *or* =%*variable*	Feedback (or reason code) that accompanies a REPORT message.	This value is used only on a REPORT. Valid values are: • NONE • Integers between MQFB_APPL_FIRST (65536) and MQFB_APPL_LAST (999999999) • %variable Corresponds to WebSphere MQ MQMD.Feedback.	MQPUT MQPUT1

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
FORMAT=*format* or =%variable	Name of the conversion exit set.	String of 1 to 8 characters. Conversion exit is invoked when the received message uses a different character set or number format than those specified in the message descriptor. Corresponds to WebSphere MQ MQMD.Format field. There is no default conversion exit set.	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
GROUPID	The GROUPID is a 24-byte field that you specify when writing messages that belong to a group. You must specify GROUPID as a null string (or spaces) on the first message of any group.	For MQGET, MQPUT, and MQPUT1 calls, the group identifier of the message is returned in the MQMD, not in the GROUPID field, which is always an input field only. On an MQGET call, this is an input field when MATCH_GROUP_ID is specified.	MQGET MQPUT MQPUT1
GSTATUS = [' ' | 'G' | 'L']	The one-byte GSTATUS field determines whether the message is part of a group. The possible values are: • 'G' (all messages in a group except the last or only one) • 'L' (the last or only logical message in a group) • Space or null (the message is not in a group)	On MQGET operations, this is an output %variable. If you are writing messages that belong to a group but are not in logical order, set this to 'L' for the message that has the highest sequence number. The default, for MQPUT and MQPUT1 operations, is null—the message does not form part of a group.	MQGET MQPUT MQPUT1

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
INPUT_AS_Q_DEF (default for local queues)	Gets messages with shared or exclusive access based on the default selected when the queue was created.	Corresponds to WebSphere MQ MQOO_INPUT_ AS_Q_DEF. Mutually exclusive with: • INPUT_EXCLUSIVE • INPUT_SHARED • BROWSE	OPEN QUEUE
INPUT_EXCLUSIVE	Gets messages and deletes them from the queue, with exclusive access to the queue.	Corresponds to WebSphere MQ MQOO_INPUT_EXCLUSI VE. Mutually exclusive with: • BROWSE • INPUT_AS_Q_DEF • INPUT_EXCLUSIVE	OPEN QUEUE
INPUT_SHARED	Gets messages and deletes them from the queue, with shared access to the queue.	Corresponds to WebSphere MQ MQOO_INPUT_SHARED. Mutually exclusive with: • BROWSE • INPUT_AS_Q_DEF • INPUT_EXCLUSIVE	OPEN QUEUE
LOCK	LOCK lets you browse a queue and lock the current message so that other threads do not see it.	The default is not to lock messages while browsing them.	MQGET
UNLOCK	UNLOCK lets you remove a lock from a previously-locked message.	When you specify UNLOCK, the message itself is not returned, it is just unlocked.	MQGET
LOGICAL_ORDER	Write one group at a time and write messages in sequential order, starting at 1.	This is the simpler way of writing a group of messages. However, it is not the default.	DEFINE QUEUE MODIFY QUEUE MQGET MQPUT OPEN QUEUE
NOT_LOGICAL_ ORDER (default)	This option gives you more control over the writing of groups, but requires that you specify SEQUENCE and GROUPID for each message.	You can also use this setting for messages that are not a group.	DEFINE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
MARK_SKIP_ BACKOUT	Messages taken from the queue under syncpoint control are not placed back on the queue if the user or the Online issues an MQBACK.	Messages are placed back on the queue, if the MQBACK is issued by the WebSphere MQ system software. Corresponds to WebSphere MQ MQGMO_MARK_SKIP_ BACKOUT. This option can be useful to prevent get, error, back out loops. See “Messages that cause errors” on page 36.	MQGET
MATCH_CORREL_ID (default)	Retrieve messages that match the specified CORRELID.	If CORRELID is not specified, this option is ignored.	MQGET
NOT_MATCH_ CORREL_ID	Retrieve any message in the queue, if CORRELID is specified.		MQGET
MATCH_MSG_ GROUP_ID	Retrieve messages that match the specified GROUPID.	The queue must be indexed by GROUPID.	MQGET
MATCH_MSG_ID (default)	Retrieve messages that match the specified MSGID.	If MSGID is not specified, this option is ignored.	MQGET
NOT_MATCH_ MSG_ID	Retrieve any message in the queue, if MSGID is specified.		MQGET
MATCH_MSG_ SEQ_NUMBER	Retrieve the message that matches SEQUENCE for the current group. If there is no current group, the only valid value of SEQUENCE is 1.	The queue must be indexed by GROUPID.	MQGET
MATCH_MSG_ TOKEN	Use this to retrieve messages that match the specified MSGTOKEN	The queue must be indexed by MSGTOKEN.	MQGET
MODEL=*name* or =%*variable*	Specifies the internal, Model 204 name of the local dynamic queue.	Must be set in the same statement with the DYNAMICQNAME option.	MQOPEN

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
MSG_UNDER_ CURSOR	Removes the message under the current browse position from the queue (a real MQGET).	Corresponds to WebSphere MQ MQGMO_MSG_UNDER_ CURSOR. Specify only if the queue is in both browse and input modes: cannot be specified with BROWSE_FIRST or BROWSE_NEXT.	MQGET
MSGID='*msgid* ' or =%*variable*	Message identifier to use as matching criterion.	Value is either a %variable or a string from 1-24 characters that: • You enter or • Queue manager automatically generates Corresponds to WebSphere MQ MQMD.MsgId.	MQGET MQPUT MQPUT1
MSGLEN=%*variable*	Specifies an output variable in which to return the actual message length.	Returned message length is less in the event of truncation.	MQGET
MSGTOKEN	You can select messages by 16-byte message token (MSGTOKEN) by specifying this field as an input field on MQGET statements with MATCH_MSG_TOKEN.	This is also an output field, retrieving the message token that was assigned by the queue manager. (On the MQGET call, it can also be an input field, if MATCH_MSG_TOKEN is specified.	MQGET MQPUT MQPUT1

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
MSGTYPE =DATAGRAM (default) *or* =REPLY *or* =REPORT *or* =REQUEST or =%variable or = application-defined numeric value	Specifies the type of message; sets the MQMD.MsgType.	Types specified by keyword are: • DATAGRAM, a simple message that needs no reply; corresponds to WebSphere MQ MQMT_DATAGRAM constant. • REPLY, a response to REQUEST; corresponds to WebSphere MQ MQMT_REPLY constant. • REPORT, a message that signifies an error condition or exception that is often requested in the report field of another message; corresponds to WebSphere MQ MQMT_REPORT constant. • REQUEST, a message that requires a reply; corresponds to WebSphere MQ MQMT_REQUEST constant.	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
NEW_CORREL_ID	This option on the MQPUT statement tells the queue manager to generate a new correlation ID for the message.	The CORRELID option should not be specified with this option, because the queue manager generates a unique correlation identifier, rather than taking it from the CORRELID option.	MQPUT MQPUT1

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
NO_CONTEXT	Message is placed in the queue without any context information.	Corresponds to WebSphere MQ option MQPMO_NO_CONTEXT. Mutually exclusive with: • DEFAULT_CONTEXT • PASS_ALL_CONTEXT • PASS_IDENTITY_ CONTEXT • SET_ALL_CONTEXT • SET_IDENTITY_ CONTEXT	MQPUT MQPUT1
NO_CONVERT	Specifies to turn off MQGMO_CONVERT.	Used in error-handling code to remove messages that did not convert from within the SOUL application and are now stuck on the queue. See “Removing messages that do not convert” on page 36.	MQGET
OUTPUT (default for remote queues)	Puts messages.	Corresponds to WebSphere MQ option MQOO_OUTPUT.	OPEN QUEUE
PASS_ALL_ CONTEXT	All context information is passed through from an input queue.	Corresponds to the WebSphere MQ option MQPMO_PASS_ALL_ CONTEXT. Mutually exclusive with: • DEFAULT_CONTEXT • NO_CONTEXT • PASS_IDENTITY_ CONTEXT • SET_ALL_CONTEXT • SET_IDENTITY_ CONTEXT	MQPUT MQPUT1

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
PASS_IDENTITY_ CONTEXT	Identity context information is passed through from an input queue.	Corresponds to the WebSphere MQ option MQPMO_PASS_IDENTIT Y_CONTEXT. Mutually exclusive with: • DEFAULT_CONTEXT • NO_CONTEXT • PASS_ALL_CONTEXT • SET_ALL_CONTEXT • SET_IDENTITY_ CONTEXT	MQPUT MQPUT1
PASS_USER_ CONTEXT	Passes user context from the queue identified in the CONTEXT parameter.	UserIdentifier field value from the last MQGET is passed along with the ALTERNATE_ USER_AUTHORITY option. The queue is opened according to the authority granted.	MQPUT1 OPEN QUEUE
PERSISTENT	Persistent message that survives a restart of the queue manager.	Corresponds to the WebSphere MQ option MQPER_ PERSISTENT of the field MQMD.Persistence. This option is an alternate to NOT_PERSISTENT and PERSISTENCE_AS_Q_D EF.	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
NOT_PERSISTENT	Message that does not survive restart of the queue manager.	Corresponds to the WebSphere MQ option MQPER_NOT_ PERSISTENT of the field MQMD.Persistence. This option is an alternate to PERSISTENT and PERSISTENCE_AS_Q_D EF.	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
PERSISTENCE_AS_ Q_DEF (default)	Persistence of the message defaults to the persistence selected when the queue was defined.	Corresponds to the WebSphere MQ option PERSISTENCE_AS_ Q_DEF of the field MQMD.Persistence. This option is tan alternate to PERSISTENT and NOT_PERSISTENT.	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
PRIORITY=*n* or PRIORITY_AS_Q_DEF (default) or =%variable	Priority of the message in the range from 0 to 9; zero is the lowest priority.	Default is priority for the queue object as defined within WebSphere MQ. Corresponds to MQMD.Priority field in a WebSphere MQ control block.	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
PUTAPPLNAME =*'value'* or =%v*ariable*	Sets the value of the PutApplName portion of the origin context.	String from 1 to 28 characters long.	MQPUT MQPUT1
PUTAPPLTYPE =*keyword* or =*number* or =%*variable*	Identifies the type of application that put the message.	If the value is a %variable or a number, it must be within the range -1 to 999,999,999 inclusive. Otherwise, the value is one of the following keywords: • AIX • CICS • CICS_VSE • DEFAULT • DOS • GUARDIAN • IMS • IMS_BRIDGE • z/OS • NO_CONTEXT • OS2 • OS400 • QMGR • UNIX • UNKNOWN • VMS • VOS • WINDOWS • WINDOWS_NT • XCF	MQPUT MQPUT1
PUTDATE=*value* or =%*variable*	Sets the value of the PutDate portion of the origin context.	String from 1 to 8 characters long.	MQPUT MQPUT1

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
PUTTIME=*value* or =%*variable*	Sets the value of the PutTime portion of the origin context.	String from 1 to 8 characters long.	MQPUT MQPUT1
REPLY_QMGR =*name* or =*%variable*	Name of queue manager for remote queue to place the reply in.	Used only on MSGTYPE=REQUEST No default. Sets the value of MQMD.ReplyToMGR. String of up to 48 characters.	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
REPLY_QUEUE =*name* *or =%variable*	Name of the queue to place the reply in.	Used only on a MSGTYPE=REQUEST. Default is the name of the queue that contained the request. Sets the values of MQMD.ReplyToQ. Set the value of MQMD.ReplyToMGR if no REPLY_QMGR option is indicated. String of up to 48 characters.	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
REPORT =*option*+*option*… or =*n* or =*%variable* (numeric values only)	Report options, which specify: • What conditions you want reported • How to construct the report message	Options are keywords that you combine with plus signs. The available option keywords are the MQRO_ options. • Keywords cannot be specified in a %variable. • Keywords are specified without the MQRO_ prefix. Alternatively, you can specify a numeric value, either as a literal or a %variable. See “Additional documentation” in the Preface for the recommended WebSphere MQ documentation, which describes the acceptable numeric values. No default report options are passed.	DEFINE QUEUE MODIFY QUEUE MQPUT MQPUT1 OPEN QUEUE
SAVE_ALL_ CONTEXT	Saves context information when messages are retrieved with MQGET.	Corresponds to WebSphere MQ MQOO_SAVE_ALL_ CONTEXT. Can be specified only if one of the INPUT_* options is also specified.	OPEN QUEUE

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
SEQUENCE =%variable	For messages that are part of a group and NOT_LOGICAL_ORDER is specified (or is the default), this number specifies the logical position of the message within the group, which need not be the physical sequence in which the messages are written. The logical first message is message sequence 1. On MQGET operations, this is an input variable used only with the MATCH_MSG_SEQ_ NUMBER option. On MQPUT and MQPUT1 operations, this is an input variable that specifies the logical position within the group of the message being written.	This is a fixed %variable. To retrieve the logical sequence of a message in an MQGET operation, use the DESCRIPTOR option and get the sequence number from the returned MQMD image. When using SEQUENCE with the MATCH_MSG_SEQ_ NUMBER option, you can only match a SEQUENCE greater than 1 within the current group. You cannot, for example, do a repeated BROWSE_NEXT with SEQUENCE 2 to jump from one group to another.	MQGET MQPUT MQPUT1
SET_ALL_CONTEXT	Origin context information is set with values from the APPLORIGINDATA, PUTAPPLNAME, PUTAPPLTYPE, PUTDATE, and PUTTIME options.	Corresponds to the WebSphere MQ option MQPMO_SET_ALL_ CONTEXT. Mutually exclusive with: • DEFAULT_CONTEXT • NO_CONTEXT • PASS_ALL_CONTEXT • PASS_IDENTITY_ CONTEXT • SET_IDENTITY_ CONTEXT	MQPUT MQPUT1

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
SET_IDENTITY_ CONTEXT	Identity context information is set with values from the ACCOUNTINGTOKEN, APPLIDENTITYDATA, and USERIDENTIFIER fields.	Corresponds to the WebSphere MQ option MQPMO_SET _IDENTITY_CONTEXT. Mutually exclusive with: • DEFAULT_CONTEXT • NO_CONTEXT • PASS_ALL_CONTEXT • PASS_IDENTITY_ CONTEXT • SET_ALL_CONTEXT	MQPUT MQPUT1
SYNCPOINT (default)	Get and put operations are under transaction control.	Operates on updates for MQCMIT and MQBACK.	DEFINE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
NO_SYNCPOINT	Get and put operations are *not* under transaction control.	Does not operate on updates for MQCMIT and MQBACK.	DEFINE QUEUE MODIFY QUEUE MQGET MQPUT MQPUT1 OPEN QUEUE
SYNCPOINT_IF_ PERSISTENT	This option on the MQGET statement works like SYNCPOINT, but is effective only for persistent messages.		MQGET
USERIDENTIFIER ='value' or =%variable	Sets the value of the User identifier portion of the identity context.	String from 1 to 12 characters long.	MQPUT MQPUT1
WAIT	Wait, if a message is unavailable.	You can displace a wait. This option is the alternate to NO_WAIT option. The maximum waiting time is specified with WAIT_TIME.	DEFINE QUEUE MODIFY QUEUE MQGET OPEN QUEUE

Table 5-1. Options for MQ/204 commands and statements (continued)

Option	Purpose	Discussion	Can be set on…
NO_WAIT (default)	Do not wait, if a message is unavailable.	Corresponds to the WebSphere MQ option MQGMO_NO_WAIT.	DEFINE QUEUE MODIFY QUEUE MQGET OPEN QUEUE
WAIT_TIME=*n* or =UNLIMITED or =%variable	Maximum time to wait, in milliseconds, if the WAIT option is specified.	Default is 0 milliseconds. To wait forever, specify a value of UNLIMITED, which corresponds to WebSphere MQ MQWI_UNLIMITED.	DEFINE QUEUE MODIFY QUEUE MQGET OPEN QUEUE

6

MQ/204 Functions Reference

Functions make it possible to reply to requests that specify a reply to queue and/or a reply to queue manager, and to aid in manipulating the MQ/204 entities. Since Model 204 entity names do not have to match their external or system names, the following functions enable the user to find out the external name of a Model 204 message queue or queue manager.

$BUFFER_ functions

These SOUL functions return the current position, size, and use of the Universal Buffer for individual users:

- $BUFFER_POSITION (formerly $MQ_BUFFER_POSITION)
- $BUFFER_SIZE (formerly $MQ_BUFFER_SIZE)
- $BUFFER_USED (formerly $MQ_BUFFER_USED)

Code that uses the former function names with the MQ prefix has been retained for compatibility with previous versions.

For more information about these functions, refer to the following Model 204 documentation wiki pages:

http://m204wiki.rocketsoftware.com/index.php/$Buffer_Position

http://m204wiki.rocketsoftware.com/index.php/$Buffer_Size

http://m204wiki.rocketsoftware.com/index.php/$Buffer_Used

http://m204wiki.rocketsoftware.com/index.php/Images

$MQ_FIND_QUEUE_ENTITY function

Function Returns the name of the queue entity that is associated with the reply to queue and reply to queue manager. You can use this function when a request

message specifies a reply to queue manager and a reply to queue; these are fields in the MQMD data structure.

Alias $MQ_FIND_Q_ENTITY

Syntax
```
$MQ_FIND_QUEUE_ENTITY(extqueueman | %qvariable,
   extqueue | %variable)
```

Where
- *extqueueman* or *%qvariable* is the external name of a queue manager, which can be a literal string enclosed in single quotation marks or a %qvariable.
- *extqueue* or *%variable* is the external name of a queue. which can be a literal string enclosed in single quotation marks or a %variable.

Example
```
%QUEUENAME = $MQ_FIND_QUEUE_ENTITY('extqueueman', 'ext-
queue'
IF (%QUEUENAME = '') THEN
   PRINT 'SOME ERROR MESSAGE'
            or
%QUEUENAME = $MQ_FIND_Q_ENTITY('extqueueman', 'extqueue')
IF (%QUEUENAME = '') THEN
   PRINT 'SOME ERROR MESSAGE'
```

Usage $MQ_FIND_QUEUE_ENTITY function determines if a queue is defined for the specified queue and queue manager combination:

- If a queue is defined, $MQ_FIND_QUEUE_ENTITY returns its name.
- If no such queue has been defined, then a 0-length string is returned.
- If multiple queue entities are mapped to the specified combination, the first one found that the user is accessing is returned, based on searching the entities in the order that they were defined.
- If the user is not accessing any of them, then the first matching entity found is returned, based on searching them in the order of definition.

$MQ_FIND_QUEUEMANAGER_ENTITY function

Function Returns the name of the queue manager entity. Because the queue manager names are defined by the system manager, they might not be known to the application developers, who need to know them to code them on MQCMIT or MQBACK.

Alias $MQ_FIND_QM_ENTITY

Syntax
```
$MQ_FIND_QUEUEMANAGER_ENTITY(queueentname | %variable)
```

Where *queueentname* or %*variable* is the name of a queue to look up, as a literal string enclosed in single quotation marks or a %variable.

Example `%QUEUEMANAGERNAME=$MQ_FIND_QUEUEMANAGER_ENTITY('queueentname')`

or

`%QUEUEMANAGERNAME=$MQ_FIND_QM_ENTITY('queueentname')`

Usage The name of the specified queue entity is looked up:

- If the queue does not exist, a null string is returned.
- If the queue exists, the name of its queue manager is returned.

$MQ_LAST_QUEUEMANAGER_ENTITY function

Function Returns the name of the queue manager most recently accessed.

Alias $MQ_LAST_QM_ENTITY

Syntax `$MQ_LAST_QUEUEMANAGER_ENTITY`

Example `%QUEUEMANAGERNAME=$MQ_LAST_QUEUEMANAGER_ENTITY`

or

`%QUEUEMANAGERNAME=$MQ_LAST_QM_ENTITY`

Usage If no queue manager has yet been accessed, the function returns a 0-length string. This is most useful after a statement that processes multiple queue managers returns an error (MQCMIT, MQBACK).

$MQ_MESSAGE_LEN function

Function Returns the actual message length of the last MQGET from or MQPUT or MQPUT1 to WebSphere MQ.

Syntax `$MQ_MESSAGE_LEN`

Where Message length can be greater than the size of the BUFFER area (if an MQGET returned a truncated message). This is the same as the value returned in MSGLEN=%mvar on an MQGET statement.

Usage Although WRITE IMAGE ON BUFFER changes the contents of the BUFFER area, it does not involve an interaction with WebSphere MQ, and hence does not change the value of $MQ_MESSAGE_LEN.

If…	Value of $MQ_MESSAGE_LEN is…
MQ/204 is not linked in	-1
At the start of a request	-1
At the start of a request running in an APSY subsystem with AUTOCOMMIT=NO	Same as the value at the end of the previous request (or -1, if this is the first request).
Before a user performs any MQGET or MQPUT/MQPUT1	-1
MQGET or MQPUT/MQPUT1	Set by the statement.
Other statements or commands	Unchanged.
Last message failed	0

$MQ_PENDING_UPDATES function

Function Returns a Boolean result indicating whether a user has made SYNCPOINT updates (controlled by MQBACK and MQCMIT statements) to a specified queue manager.

Syntax

```
IF ($MQ_PENDING_UPDATES('queuemanentname' | %variable))
THEN
```

or

```
IF ($MQ_PENDING_UPDATES) THEN
```

Where *queuemanentname* is a literal string enclosed in single quotation marks or *%variable* specifying the name of a queue manager.

Usage If the user has done syncpoint gets or puts that have not yet been committed or backed out, the function returns true (1), otherwise it returns false (0).

If no argument is specified, then the Boolean result returned covers all queue managers being accessed.

$MQ_QUEUENAME function

Function Returns the external queue name for a specified queue.

Alias $MQ_QNAME

Syntax `%REALNAME=$MQ_QUEUENAME('queueentname' | %variable)`

Where *queueentname* is a literal string enclosed in single quotation marks or a %variable

Usage The $MQ_QUEUENAME function can be useful for:

- Reporting errors
- Retrieving the queue name that WebSphere MQ assigned to a local dynamic queue when the input dynamicq ends with an asterisk (*) for the OPEN QUEUE statement.

$MQ_QUEUEMANAGERNAME function

Function Returns the external queue manager name for a specified queue manager.

Alias $MQ_QMNAME('*queuemanentname'* | *%variable*)

Syntax `%REALNAME=$MQ_QUEUEMANAGERNAME('queuemanentname' | %variable)`

Where *queuemanentname* is a literal string enclosed in single quotation marks or a %variable.

Usage The $MQ_QUEUEMANAGERNAME function can be useful for error reporting.

7

Configuring MQ/204 for a Windows NT PC

You can use the sample files described in this chapter to successfully test the connections between a Windows NT PC running WebSphere MQ for Windows and z/OS platform running MQ/204 and Model 204.

Preinstallation

The files are part of the MQ/204 product distribution. Relocate the following files to the following locations:

File name	Put the file in…
MQz/OS.TXT	
MQNTMQD.TXT	C:\Program Files \ WebSphere MQ for Windows \ Createmq.mqd
MQNTTST.TXT	C:\Program Files \ WebSphere MQ for Windows \ Samples
MQULINIT.TXT	
MQULPUT.TXT	
MQULGET.TXT	
z/OSTRACT.TXT	

Making the files site-specific

The sample files that accompany this chapter use:

- CSQ5 as the identifier for the mainframe z/OS queue manager name. If the queue manager on the mainframe at your site has a different name, change

all occurrences of CSQ5 to your site's queue manager name in each sample file.

- PLUTO as the name of the MQ/204 queue manager on the NT machine. Rocket recommends that you accept this name to avoid changing it in many places.

Installation considerations

Configuration requirements

The following software must be installed:

- Model 204 V5.1 or higher
- MQ/204
- WebSphere MQ for z/OS
- WebSphere MQ for Windows, on a Windows NT computer

Starting WebSphere MQ queue manager

You must include the MQ.SCSQAUTH library in the STEPLIB concatenation to successfully start the queue manager. When //STEPLIB does not include a reference to the IBM WebSphere MQ libraries, Model 204 displays the following messages:

```
M204.2506: function USER=user-id COMP_CODE=completion-
code RSN_CODE=reason-code QM=queue-manager EXT_QM=exter-
nal-name TASK=mqtask-address

M204.2543: CONNECTION TO QUEUEMANAGER queue-manager-name
FAILED, REASON CODE reason-code

M204.2544: UNABLE TO START QUEUEMANAGER
```

Concatenate the following libraries with your STEPLIB when executing the ONLINE Model 204 module.

JCL to concatenate libraries

```
//STEPLIB  DD  DISP=SHR,DSN=Your-favorite-V4.2-LOADLIB
//         DD  DSN=MQSERIES.SCSQAUTH,DISP=SHR
//         DD  DSN=MQSERIES.SCSQANLE,DISP=SHR
```

CCAIN settings for MQ/204

```
MQINTASK=010,MQMXTASK=150,MQWAIT=300000
```

CSQUTIL sample JCL

```
//*                          PARM='QUEUE MANAGER NAME'
//CSQUTIL EXEC PGM=CSQUTIL,PARM='CSQ5'
//STEPLIB  DD  DSN=MQSERIES.SCSQAUTH,DISP=SHR
//         DD  DSN=MQSERIES.SCSQANLE,DISP=SHR
//SYSPRINT DD  SYSOUT=W
//OUTDEF   DD  SYSOUT=S
//SYSIN    DD *
COMMAND DDNAME(SYSIN2) MAKEDEF(OUTDEF)
/*
//SYSIN2  DD *
DELETE QLOCAL (CLIQUE1) PURGE
DEFINE QLOCAL (CLIQUE1)
/*
```

Configuring WebSphere MQ queue manager to MQ/204

The CSQ5.z/OS file contains the sample z/OS definitions (internal reference MQz/OS PLUTO) for:

- QLOCAL (’CSQ5.XMIT.TO.PLUTO’)
- PROCESS (’CSQ5.PLUTO.PROCESS’)
- QREMOTE (’PLUTO.BOUND.QUEUE.REMOTE’)
- CHANNEL (’CSQ5.TO.PLUTO’)
- CHANNEL (’PLUTO.TO.CSQ5’)
- QLOCAL (’CSQ5.BOUND.QUEUE.LOCAL’)

You must run these DEFINE commands against that queue manager on the z/OS instance of WebSphere MQ that Model 204 connects to.

DEFINE commands for MQ/204 queue manager

```
*********************************************************************
*   MQz/OS   PLUTO - You might want to make this file a member   *
*   of the CSQ5.SCSQPROC PDS named PLUTO.                          *
*********************************************************************
*                                                                   *
*   You must change the value of the CONNAME argument in the       *
*   DEFINE CHANNEL commands to use the TCP/IP address and port or  *
*   if the default port is 1414, the IP address only, on which the *
*   PC NT WebSphere MQ for Windows queue manager is running.       *
*                                                                   *
*********************************************************************
*                                                                   *
*   Run these DEFINE commands against queue manager CSQ5.          *
*                                                                   *
```

```
************************************************************************
DEFINE QLOCAL('CSQ5.XMIT.TO.PLUTO') REPLACE +
       DESCR('Local transmission queue')  +
       PUT(ENABLED) GET(ENABLED) TRIGGER TRIGTYPE(FIRST) +
       USAGE(XMITQ) INITQ('SYSTEM.CHANNEL.INITQ') +
       PROCESS('CSQ5.PLUTO.PROCESS')
DEFINE PROCESS( 'CSQ5.PLUTO.PROCESS' ) REPLACE +
         DESCR( 'Process for sending messages to PLUTO' ) +
         APPLTYPE( z/OS ) +
         APPLICID( 'CSQX START' ) +
         USERDATA( 'CSQ5.TO.PLUTO' ) +
         ENVRDATA( ' ' )
DEFINE QREMOTE('PLUTO.BOUND.QUEUE.REMOTE') REPLACE  +
       DESCR('Remote queue defined on CSQ5') +
       DEFPSIST(YES) +
       RNAME('PLUTO.BOUND.QUEUE.LOCAL') +
       RQMNAME('PLUTO') +
       XMITQ('CSQ5.XMIT.TO.PLUTO')
DEFINE CHANNEL ('CSQ5.TO.PLUTO')  CHLTYPE(SDR) TRPTYPE(TCP) +
       XMITQ('CSQ5.XMIT.TO.PLUTO') +
       CONNAME('<PUT PC NT TCPIP ADDRESS HERE>') +
       DISCINT(0) +
       DESCR('Sender channel for messages to queue manager PLUTO') +
       REPLACE
DEFINE CHANNEL ('PLUTO.TO.CSQ5') CHLTYPE(RCVR) TRPTYPE(TCP) +
      DESCR('Requester channel for messages from queue manager PLUTO')
+
       REPLACE
DEFINE QLOCAL('CSQ5.BOUND.QUEUE.LOCAL') REPLACE +
       DESCR('Local queue') +
       DEFPSIST(YES)  +
       SHARE
```

Where You supply the site-specific PC NT TCP/IP address.

Configuring WebSphere MQ for Windows

The PLUTO.MQD file contains the sample NT definitions for:

```
ComponentType=QueueManager
Name=PLUTO
ComponentType=ChannelGroup
Name=PLUTOGroup
QueueManagerName=PLUTO
ComponentType=Connection
Name=PLUTO_Connection
QueueManagerName=PLUTO
ChannelGroupName=PLUTOGroup
```

After you make a backup copy of the "C:\Program Files\WebSphere MQ for Windows\Createmq.mqd" PC file, replace its contents with those of this file. When you reboot the NT machine, WebSphere MQ for Windows notices the new Createmq.mqd file, and prompts you by asking if you want to replace *the-earlier-version*.

Select the YES option.

File to run WebSphere MQ

```
*********************************************************************
*   MQNTMQD PLUTO - Place this file on PC as                        *
*   C:\Program Files\WebSphere MQ for Windows\Createmq.mqd          *
*   after saving a copy of the original file.                       *
*   Reboot the PC to cause WebSphere MQ to run the file.            *
*********************************************************************
[Component_1]
ComponentType=QueueManager
Name=PLUTO
Description=Queue manager to communicate with CSQ5
LoadUserMQSC_1=\Program Files\WebSphere MQ for Windows\Sam-
ples\PLUTO.tst
Replace=yes

[Component_2]
ComponentType=ChannelGroup
Name=PLUTOGroup
Description=Channel group to communicate with CSQ5
QueueManagerName=PLUTO
AllUserChannels=no
StartListener=yes
Channel_1=PLUTO.TO.CSQ5
Channel_2=CSQ5.TO.PLUTO
Replace=yes

[Component_3]
ComponentType=Connection
Name=PLUTO_Connection
Description=Connection to use on PLUTO machine
QueueManagerName=PLUTO
HasChannelGroup=yes
ChannelGroupName=PLUTOGroup
Replace=yes
Autostart=yes
```

Configuring TCP/IP

The PLUTO.TST file contains the sample NT definitions (internal reference MQNTTST PLUTO) for:

- QLOCAL (’PLUTO.XMIT.TO.CSQ5’)
- QREMOTE (’CSQ5.BOUND.QUEUE.REMOTE’)
- CHANNEL (’PLUTO.TO.CSQ5’)
- CHANNEL (’CSQ5.TO.PLUTO’)
- QLOCAL (’PLUTO.BOUND.QUEUE.LOCAL’)

These definitions must be run on the NT WebSphere MQ for Windows that you are testing. You may find it convenient to place this file in the "C:\Program Files\WebSphere MQ for Windows\Samples” directory where other samples are found. Start the PLUTO queue manager and run this MQSC file; the PC is then be ready to conduct the test.

Usage notes When the test application places messages on a queue on PLUTO, the NT PC, you must place the messages on the queue, CSQ5.BOUND.QUEUE.REMOTE. WebSphere MQ for Windows moves the message to the PLUTO.XMIT.TO.CSQ5 queue for transmission.

After WebSphere MQ transmits the message, you can find it on the z/OS side CSQ5.BOUND.QUEUE.LOCAL queue. You can use the SOUL program MQGET to retrieve this message. (See “Retrieving data from a queue” on page 119.)

In the following test application you need to supply only your IP address in the first DEFINE CHANNEL command that includes the CONNAME argument.

DEFINE commands for TCP/IP

```
*********************************************************************
*   MQNTTTST PLUTO - You may find it convenient to place this file *
*   in the "C:\Program Files\WebSphere MQ for Windows\Samples"     *
*   directory where other samples are found.                       *
*                                                                  *
*   You must change the value of the CONNAME argument in the       *
*   channel definition (DEFINE CHANNEL commands) to use the TCP/IP *
*   address and port, or if the default port is 1414, the IP       *
*   address only, on which the CSQ5 queue manager is running.      *
*********************************************************************
*                                                                   *
*   Run this sample MQSC file from queue manager PLUTO.             *
*                                                                   *
*********************************************************************
DEFINE QLOCAL(’PLUTO.XMIT.TO.CSQ5’) REPLACE +
```

```
        PUT(ENABLED) GET(ENABLED) +
        INITQ('SYSTEM.CHANNEL.INITQ') +
        DESCR('Local transmission to CSQ5 queue')  +
        USAGE(XMITQ)
DEFINE QREMOTE('CSQ5.BOUND.QUEUE.REMOTE') REPLACE  +
        DESCR('Remote queue defined on PLUTO') +
        DEFPSIST(YES) +
        RNAME('CSQ5.BOUND.QUEUE.LOCAL') +
        RQMNAME('CSQ5') +
        XMITQ('PLUTO.XMIT.TO.CSQ5')
DEFINE CHANNEL ('PLUTO.TO.CSQ5')  CHLTYPE(SDR) TRPTYPE(TCP) +
        XMITQ('PLUTO.XMIT.TO.CSQ5') +
        CONNAME('<PUT CSQ5 TCPIP ADDRESS HERE>') +
        DESCR('Sender channel for messages to queue manager CSQ5') +
        REPLACE
DEFINE CHANNEL ('CSQ5.TO.PLUTO') CHLTYPE(RCVR) TRPTYPE(TCP) +
        DESCR('Receiver channel for messages from queue manager CSQ5') +
        REPLACE
DEFINE QLOCAL('PLUTO.BOUND.QUEUE.LOCAL') REPLACE +
        DESCR('Local queue for messages from CSQ5') +
        DEFPSIST(YES)  +
        SHARE
```

Where You supply the CSQ5 TCP/IP address in the previous DEFINE CHANNEL command.

Initializing WebSphere MQ for Windows

The ULINIT.TXT file contains the sample definitions (internal reference MQULINIT PLUTO) and the test.

```
* MQPUT TEST TO COMMUNICATE WITH WebSphere MQ for Windows on PLUTO
* INITIALIZATION
DEFINE QM WELQM5   WITH SCOPE=SYSTEM QMNAME=CSQ5
DEFINE Q  CREMOTE  WITH SCOPE=SYSTEM QM=WELQM5
QNAME=PLUTO.BOUND.QUEUE.REMOTE
DEFINE Q  CLOCAL   WITH SCOPE=SYSTEM QM=WELQM5
QNAME=CSQ5.BOUND.QUEUE.LOCAL
START QM WELQM5
MONITOR MQ SUBTASKS
```

Putting data on a queue

The ULPUT.TXT file contains a sample MQPUT program that reads a procedure, translates it into ASCII, and places the messages on the PLUTO.BOUND.QUEUE.REMOTE.

From there, WebSphere MQ moves them to the ('CSQ5.XMIT.TO.PLUTO') queue for transmission. When they arrive on PLUTO, the messages are in the PLUTO.BOUND.QUEUE.LOCAL. You can browse the messages using the

AMQSBCGW.EXE sample application, or you can retrieve them using the AMQSGETW.EXEC application.

These applications are found in the C:\Program Files\WebSphere MQ for Windows\Samples\C\ directory

```
* MQPUT TEST TO COMMUNICATE WITH WebSphere MQ for Windows on PLUTO
BEGIN
%PROC IS STRING LEN 255
PRINT 'ENTER FILENAME AND PROCEDURE NAME TO XMIT TO PLUTO'
%FILE = '??FILENAME'
%PROC = '??PROCNAME'
SUBROUTINE PRINT.STATUS:
IF $STATUS NE 0 OR $STATUSD NE 0 THEN
 PRINT '$STATUS/$STATUSD = ' WITH $STATUS WITH '/' WITH $STATUSD
 SKIP 1 LINE
END IF
RETURN
END SUBROUTINE
*** MAINLINE ***
%TEXT IS STRING LEN 100
PRINT 'ABOUT TO OPEN QUEUE ...'
OPEN QUEUE CREMOTE OUTPUT
CALL PRINT.STATUS
* READ THE PROC AND SEND EACH LINE TO PLUTO
PRINT 'ATTEMPTING TO SEND ' %PROC ' IN ' %FILE
         %CTLID = $RDPROC('OPEN',%FILE,%PROC)
         IF $STATUS THEN
             JUMP TO BAILOUT
         END IF
         %RDSTAT = $STATUS
         REPEAT WHILE NOT %RDSTAT
             %TEXT = $RDPROC('GET',%CTLID)
             %RDSTAT = $STATUS

*PRINT 'ABOUT TO ADD FOLLOWING MESSAGE TO THE QUEUE'
 PRINT %TEXT
* TRANSLATE TO ASCII FOR NT PC
 %TEXT = $ASCII(%TEXT)
 MQPUT %TEXT ON CREMOTE
 CALL PRINT.STATUS
         END REPEAT
BAILOUT:
PRINT 'COMMITING'
 MQCMIT
CALL PRINT.STATUS

PRINT 'ABOUT TO CLOSE QUEUE ...'
CLOSE QUEUE CREMOTE
CALL PRINT.STATUS
```

```
END
```

Retrieving data from a queue

The ULGET.TXT file contains a sample MQGET program that obtains messages from the CSQ5.BOUND.QUEUE.LOCAL queue, translates them from ASCII to EBCDIC, and displays them on the terminal or writes them to the USE data set.

To test getting messages on the CSQ5.BOUND.QUEUE.LOCAL queue, run the AMQSPUTW.EXE application in the C:\Program Files\WebSphere MQ for Windows\Samples\C\ directory on the NT PC. (If you do this locally, you are not testing the configuration.) With this program put one or more messages on the CSQ5.BOUND.QUEUE.REMOTE. From here, WebSphere MQ moves the messages to the ('PLUTO.XMIT.TO.CSQ5') queue for transmission.

When they arrive on the CSQ5 z/OS side, run this MQGET SOUL procedure to retrieve them.

```
* MQGET TEST TO COMMUNICATE WITH WebSphere MQ for Windows on PLUTO
BEGIN
%TEXT IS STRING LEN 255
SUBROUTINE PRINT.STATUS
IF $STATUS NE 0 OR $STATUSD NE 0 THEN
 PRINT '$STATUS/$STATUSD = ' WITH $STATUS WITH '/' WITH $STATUSD
 SKIP 1 LINE
END IF
RETURN
END SUBROUTINE

*** MAINLINE ***
PRINT 'ABOUT TO OPEN QUEUE ...'
OPEN QUEUE CLOCAL
CALL PRINT.STATUS
PRINT 'ATTEMPT TO RETRIEVE MESSAGE'
%TEXT = 'NO TEXT RECEIVED'
 MQGET %TEXT FROM CLOCAL NO_WAIT
 CALL PRINT.STATUS
* text is in ascii
 PRINT $TIME WITH ' MQGET: ' WITH $EBCDIC(%TEXT)

PRINT 'ABOUT TO CLOSE QUEUE ...'
CLOSE QUEUE CLOCAL
CALL PRINT.STATUS
END
```

Trace information

The TRACE.TXT file contains information about the WebSphere MQ debugging information available on the mainframe in the Model 204 internal trace table.

Trace facility

The following levels of trace information are added to the wrap-around trace table, if one is allocated, and the MQ/204 trace options are activated. The trace options are:

```
RESET DBGBIT X'00001700'
```

which represents the separate bits:

Hex setting	Trace option	Traces
X'01'	MQ	Information identical to that included in the RK audit trail lines. Trace entries are made for all WebSphere MQ API calls, including MQCONN and MQDISC.
X'02'	MQM	MQMD (message descriptor) control block. Trace entries are made only for MQGET, MQPUT, and MQPUT1.
X'04'	MQO	For MQPUT and MQPUT1, traces the MQPMO (put message options) control block. For MQGET, traces the MQGMO (get message options) control block.
X'08'	MQD	First 100 bytes of message data. Trace entries are made only for MQGET, MQPUT, and MQPUT1.
X'10'	MQA	Entire message. Trace entries are made only for MQGET, MQPUT, and MQPUT1.

How to use the trace table in Model 204

Setting the TRACESIZ parameter

Set the TRACESIZ parameter in the CCAIN stream of the ONLINE or BATCH204 job. The value of the TRACESIZ parameter determines how many bytes of storage to allocate to the wrap around trace table:

```
TRACESIZ=50000,
X
```

Activating the trace

Choose the proper *ZAP command for the version of Model 204 that you are running to customize this component.

```
* BY Default, when Model 204 is delivered, the trace facility is dis-
abled
* These *ZAPs enable tracing
* STAR ZAP FOR VERSION V4.2.0
* ZAP PARM X'4C94' X'E55C' X'E4B4'
* STAR ZAP FOR VERSION V4.1.1
```

```
* ZAP PARM X’47D4’ X’E55C’ X’E4B4’
* STAR ZAP FOR VERSION V4.1.0
* ZAP PARM X’4520’ X’E55C’ X’E4B4’
RESET DBGBIT X’00001700’
RESET TRACEFLG 2
* Now that TRACESIZ, TRACEFLG and DBGBIT are set, MQ/204 Trace entries
should
* appear in the trace table.
```

Viewing the trace table

Once you think that you have entries in the trace table, you can examine them. The following examples illustrate viewing the trace table using *LOOK and *SNAP:

```
* Look at entries 5 through 10 in the trace table
*LOOK TRACETBL 5 10

* Look at all the entries in the trace table
*LOOK TRACETBL 0 9999 (however many 9’s needed at your
site)

* The *SNAP command prints the entire trace table
* in the formatted snap
*SNAP

* Star zap for version 4.2.1
*’Zap parm X’4C8C’ x ’E55C’ x ’E4B4’
```

Index

Symbols

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

W

Z