WebSphere MQ for z/OS

z/OS Concepts and Planning Guide

Version 7.0

GC34-6926-01

WebSphere MQ for z/OS

z/OS Concepts and Planning Guide

Version 7.0

GC34-6926-01

Note
Before using this information and the product it supports, be sure to read the general information under notices at the back of this book.

Second edition (January 2009)

This edition of the book applies to the following products:

- IBM WebSphere MQ for z/OS, Version 7.0

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

Contents

Figures

Tables

Chapter 1. Introduction

This book describes things that you need to know about WebSphere® MQ for z/OS® before you can install and use it on your z/OS system. It explains the concepts of WebSphere MQ, and gives you information to help you plan your WebSphere MQ subsystems. If you need to find out about WebSphere MQ on all platforms, see *MQSeries : An Introduction to Messaging and Queuing*.

This chapter introduces the concepts you need to understand, and directs you to more detailed explanations later in the book. It contains the following sections:

What is a message?

A *message* is a string of bytes that is meaningful to the applications that use it. Messages are used to transfer information from one application program to another (or to different parts of the same application). The applications can be running on the same platform, or on different platforms.

WebSphere MQ messages have two parts:

- The *application data*. The content and structure of the application data is defined by the application programs that use them.
- A *message descriptor*. The message descriptor identifies the message and contains additional control information, such as the type of message and the priority assigned to the message by the sending application.

What is a queue?

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

Queues are managed by a *queue manager*. The queue manager is responsible for maintaining the queues it owns, storing all the messages it receives onto the appropriate queues, and retrieving the messages in response to application requests. The messages might be put on the queues by application programs, or by a queue manager as part of its operation.

What is a WebSphere MQ object?

Many of the tasks you carry out when using WebSphere MQ involve manipulating WebSphere MQ *objects*. In WebSphere MQ for z/OS the object types are queues, processes, authentication information objects, namelists, storage classes, Coupling Facility structures, channels, clusters, system objects, and queue manager security objects.

The manipulation or administration of objects includes the following:

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

The properties of an object are defined by its attributes. Some you can specify, others you can only view.

What is a queue manager?

Before you can let your application programs use WebSphere MQ on your z/OS system, you must install the WebSphere MQ for z/OS product and start a queue manager. The queue manager owns and manages the set of resources that are used by WebSphere MQ. These resources include:

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

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

Figure 1 on page 3 illustrates a queue manager, showing connections to different application environments, and the channel initiator.

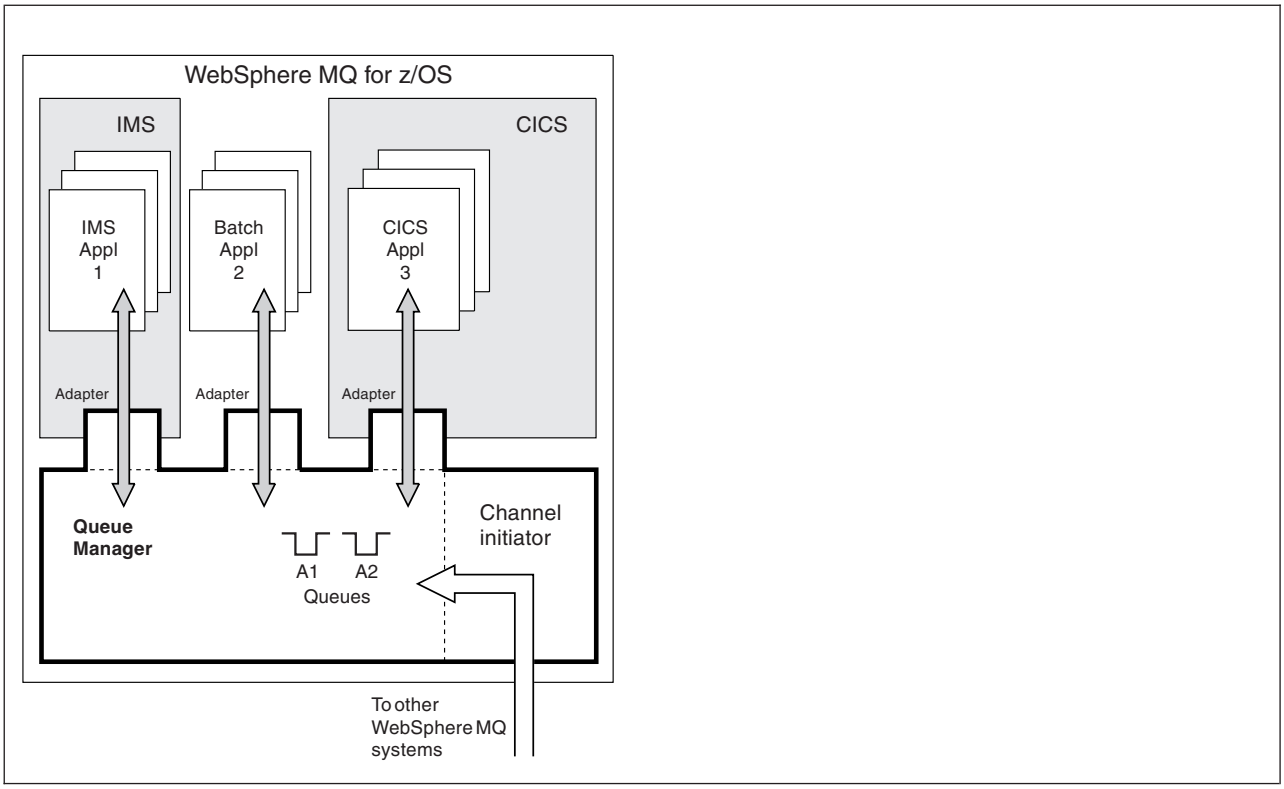


Figure 1. Overview of WebSphere MQ for z/OS

The queue manager subsystem

On z/OS, WebSphere MQ runs as a z/OS subsystem that is started at IPL time. Within the subsystem, the queue manager is started by executing a JCL procedure that specifies the z/OS data sets that contain information about the logs, and that hold object definitions and message data (the page sets). The subsystem and the queue manager have the same name, of up to four characters. All queue managers in your network must have unique names, even if they are on different systems, sysplexes, or platforms.

Shared queues

Queues can be *non-shared*, owned by and accessible to only one queue manager, or *shared*, owned by a *queue-sharing group*. A queue-sharing group consists of a number of queue managers, running within a single z/OS sysplex, that can access the same WebSphere MQ object definitions and message data concurrently. Within a queue-sharing group, the shareable object definitions are stored in a shared DB2® database, and the messages are held inside one or more Coupling Facility structures (CF structures), and in the shared DB2 database. The shared DB2 database and the Coupling Facility structures are resources that are owned by several queue managers.

Page sets and buffer pools

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

operation gets a message from the same queue. If the message is removed from the queue, space in the page set that holds the data is subsequently freed for reuse. As the number of messages held on a queue increases, so the amount of space used in the page set increases, and as the number of messages on a queue reduces, the space used in the page set reduces.

To reduce the overhead of writing data to and reading data from the page sets, the queue manager buffers the updates into processor storage. The amount of storage used to buffer the page set access is controlled through WebSphere MQ objects called *buffer pools*.

For more information about page sets and buffer pools, see 'Chapter 3 Storage Management'.

Logging

Any changes to objects held on page sets, and operations on persistent messages, are recorded as log records. These log records are written to a log data set called the *active log*. The name and size of the active log data set is held in a data set called the *bootstrap data set* (BSDS).

When the active log data set fills up, the queue manager switches to another log data set so that logging can continue, and copies the content of the full active log data set to an *archive log* data set. Information about these actions, including the name of the archive log data set, is held in the bootstrap data set. Conceptually, there is a ring of active log data sets that the queue manager cycles through; when an active log is filled, the log data is off-loaded to an archive log, and the active log data set is available for reuse.

The log and bootstrap data sets are discussed in "Logging" on page 25.

Tailoring the queue manager environment

When the queue manager is started, a set of initialization parameters that control how the queue manager operates are read. In addition, data sets containing WebSphere MQ commands are read, and the commands they contain are executed. Typically, these data sets contain definitions of the system objects required for WebSphere MQ to run, and you can tailor these to define or initialize the WebSphere MQ objects necessary for your operating environment. When these data sets have been read, any objects defined by them are stored, either on a page set or in DB2.

Initialization parameters and system objects are discussed in "Defining your system" on page 34.

Recovery and restart

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

If the queue manager terminates abnormally, the recovery phase of queue manager restart can recover the lost page set changes because persistent message data is

held in log records. This means that WebSphere MQ can recover persistent message data and object changes right up to the point of failure.

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

Recovery and restart are discussed in "Recovery and restart" on page 44.

Security

You can use an external security manager, such as Security Server (previously known as RACF®) to protect the resources that WebSphere MQ owns and manages from access by unauthorized users. You can also use the Secure Sockets Layer (SSL) for channel security. SSL is included as part of the WebSphere MQ product.

WebSphere MQ security is discussed in "Security" on page 59.

Availability

There are several features of WebSphere MQ that are designed to increase system availability in the event of queue manager or communications subsystem failure. These features are discussed in "Availability" on page 66.

Manipulating objects

When the queue manager is running, you can manipulate WebSphere MQ objects either through a z/OS console interface, or through an administration utility that uses ISPF services under TSO. Both mechanisms allow you to define, alter, or delete WebSphere MQ objects. You can also control and display the status of various WebSphere MQ and queue manager functions.

You can also manipulate WebSphere MQ objects using the WebSphere MQ Version 7 Explorer: a graphical user interface that provides a visual way of working with queues, queue managers and other objects.

These facilities are discussed in "Commands" on page 69.

Monitoring and statistics

Several facilities are available to monitor your queue managers and channel initiators. You can also collect statistics for performance evaluation and accounting purposes.

These facilities are discussed in "Monitoring and statistics" on page 79.

Application environments

When the queue manager has started, applications can connect to it and start using the WebSphere MQ API. These can be CICS, IMS, Batch, or WebSphere Application Server applications. WebSphere MQ applications can also access applications on CICS and IMS systems that are not aware of WebSphere MQ, using the CICS and IMS bridges.

These facilities are discussed in Chapter 3, "WebSphere MQ and other products," on page 83.

For information about writing WebSphere MQ applications, see the following manuals:

- WebSphere MQ Application Programming Guide
- WebSphere MQ Using C++
- WebSphere MQ Using Java

What is a channel initiator?

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

To send messages from queue manager A to queue manager B, a *sending* MCA on queue manager A must set up a communications link to queue manager B. A *receiving* MCA must be started on queue manager B to receive messages from the communications link. This one-way path consisting of the sending MCA, the communications link, and the receiving MCA is known as a *channel*. The sending MCA takes messages from a transmission queue and sends them down a channel to the receiving MCA. The receiving MCA receives the messages and puts them on to the destination queues.

In WebSphere MQ for z/OS, the sending and receiving MCAs all run inside the channel initiator (the channel initiator is also known as the *mover*). The channel initiator runs as a z/OS address space under the control of the queue manager. There can be only a single channel initiator connected to a queue manager and it is run inside the same z/OS image as the queue manager. There can be thousands of MCA processes running inside the channel initiator concurrently.

Figure 2 on page 7 shows two queue managers within a sysplex. Each queue manager has a channel initiator and a local queue. Messages sent by queue managers on AIX® and Windows® are placed on the local queue, from where they are retrieved by an application. Reply messages are returned by a similar route.

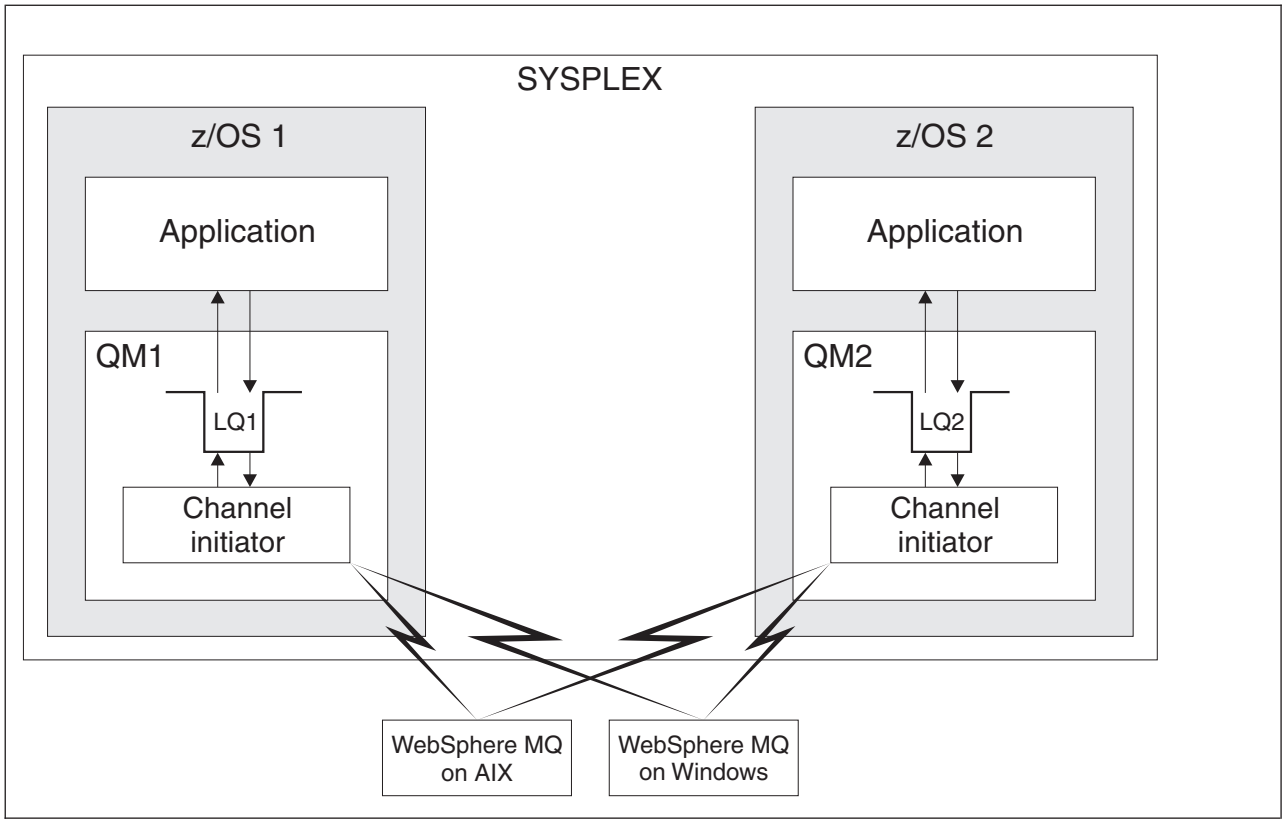


Figure 2. Communication between queue managers

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

Listeners
These listen for inbound channel requests on a communications subsystem such as TCP, and start a named MCA when an inbound request is received.

Supervisor
This manages the channel initiator address space, for example it is responsible for restarting channels after a failure.

Name server
This is used to resolve TCP names into addresses.

SSL tasks
These are used to perform encryption and decryption and check certificate revocation lists.

Distributed queuing is described in the WebSphere MQ Intercommunication manual.

Queue manager clusters

You can group queue managers in a *cluster*. Queue managers in a cluster can make the queues that they host available to every other queue manager in the cluster. Any queue manager can send a message to any other queue manager in the same cluster without the need for many of the object definitions required for standard

distributed queuing. Each queue manager in the cluster has a single transmission queue from which it can transmit messages to any other queue manager in the cluster.

Clusters are described in the WebSphere MQ Queue Manager Clusters manual.

Chapter 2. WebSphere MQ for z/OS concepts

Shared queues and queue-sharing groups

Shared queues and queue-sharing groups allow high availability of MQ resources. This describes the attributes and benefits.

This chapter describes how several queue managers can share the same queues and the messages on those queues. It discusses the following topics:

What is a shared queue?

A *shared queue* is a type of local queue. The messages on that queue can be accessed by one or more queue managers that are in a sysplex. The queue managers that can access the same set of shared queues form a group called a *queue-sharing group*.

Any queue manager can access messages

Any queue manager in the queue-sharing group can access a shared queue. This means that you can put a message on to a shared queue on one queue manager, and get the same message from the queue from a different queue manager. This provides a rapid mechanism for communication within a queue-sharing group that does not require channels to be active between queue managers.

Large messages (>63KB) have a placeholder stored in the Coupling Facility (4K), and their message data stored in DB2. Figure 3 on page 10 shows three queue managers and a Coupling Facility, forming a queue-sharing group. All three queue managers can access the shared queue in the Coupling Facility.

An application can connect to any of the queue managers within the queue-sharing group. Because all the queue managers in the queue-sharing group can access all the shared queues, the application does not depend on the availability of a specific queue manager; any queue manager in the queue-sharing group can service the queue.

This gives greater availability because all the other queue managers in the queue-sharing group can continue processing the queue if one of the queue managers has a problem.

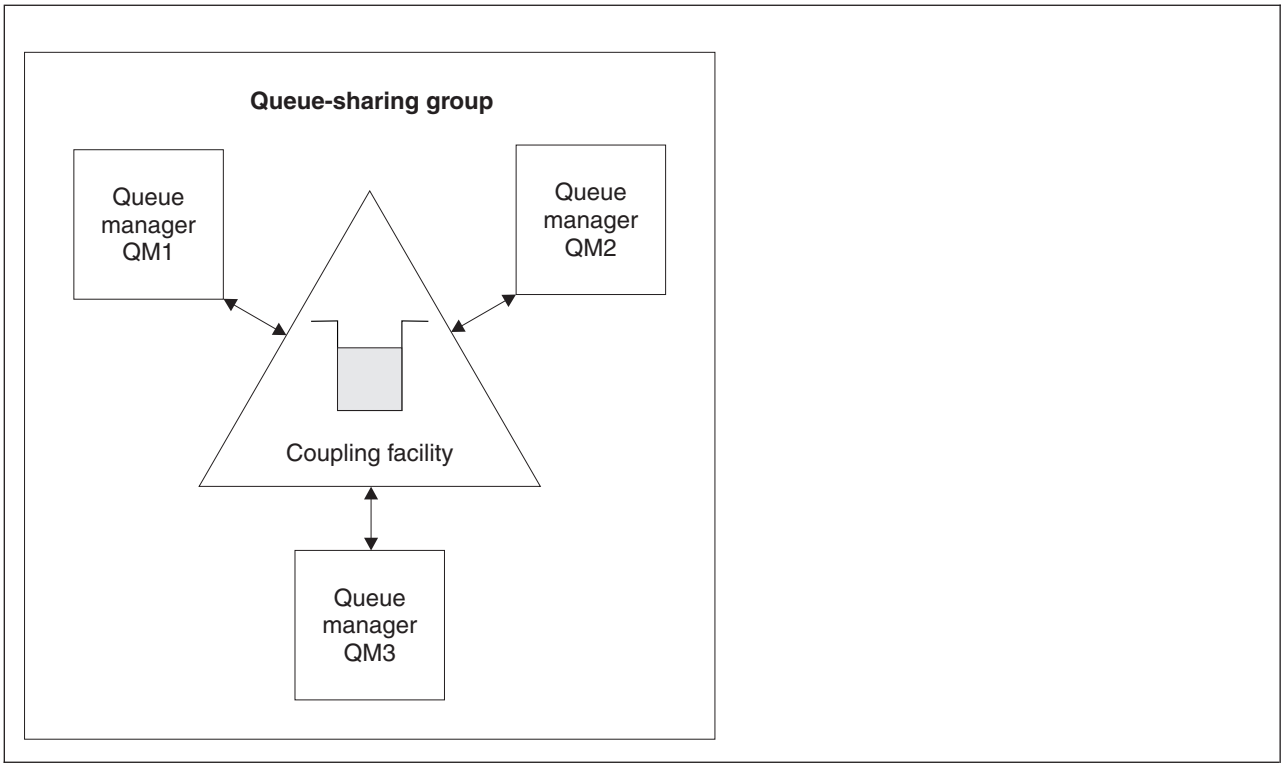


Figure 3. A queue-sharing group

Queue definition shared by all queue managers

Shared queue definitions are stored in the DB2 database table OBJ_B_QUEUE. Because of this, you need to define the queue only once and then it can be accessed by all the queue managers in the queue-sharing group. This means that there are fewer definitions to make.

By contrast, the definition of a non-shared queue is stored on page set zero of the queue manager that owns the queue (as described in "Page sets" on page 20).

You cannot define a shared queue if a queue with that name has already been defined on the page sets of the defining queue manager. Likewise, you cannot define a local version of a queue on the queue manager page sets if a shared queue with the same name already exists.

What is a queue-sharing group?

The group of queue managers that can access the same shared queues is called a queue-sharing group. Each member of the queue-sharing group has access to the same set of shared queues.

Queue-sharing groups have a name of up to four characters. The name must be unique in your network, and must be different from any queue manager names.

Figure 4 on page 11 illustrates a queue-sharing group that contains two queue managers. Each queue manager has a channel initiator and its own local page sets and log data sets.

Each member of the queue-sharing group must also connect to a DB2 system. The DB2 systems must all be in the same DB2 data-sharing group so that the queue managers can access the DB2 shared repository used to hold shared object definitions. These are definitions of any type of WebSphere MQ object (for example, queues and channels) that are defined only once and then any queue manager in the group can use them. These are called *global* definitions and are described in "Private and global definitions" on page 70.

More than one queue-sharing group can reference a particular data-sharing group. You specify the name of the DB2 subsystem and which data-sharing group a queue manager uses in the WebSphere MQ system parameters at startup.

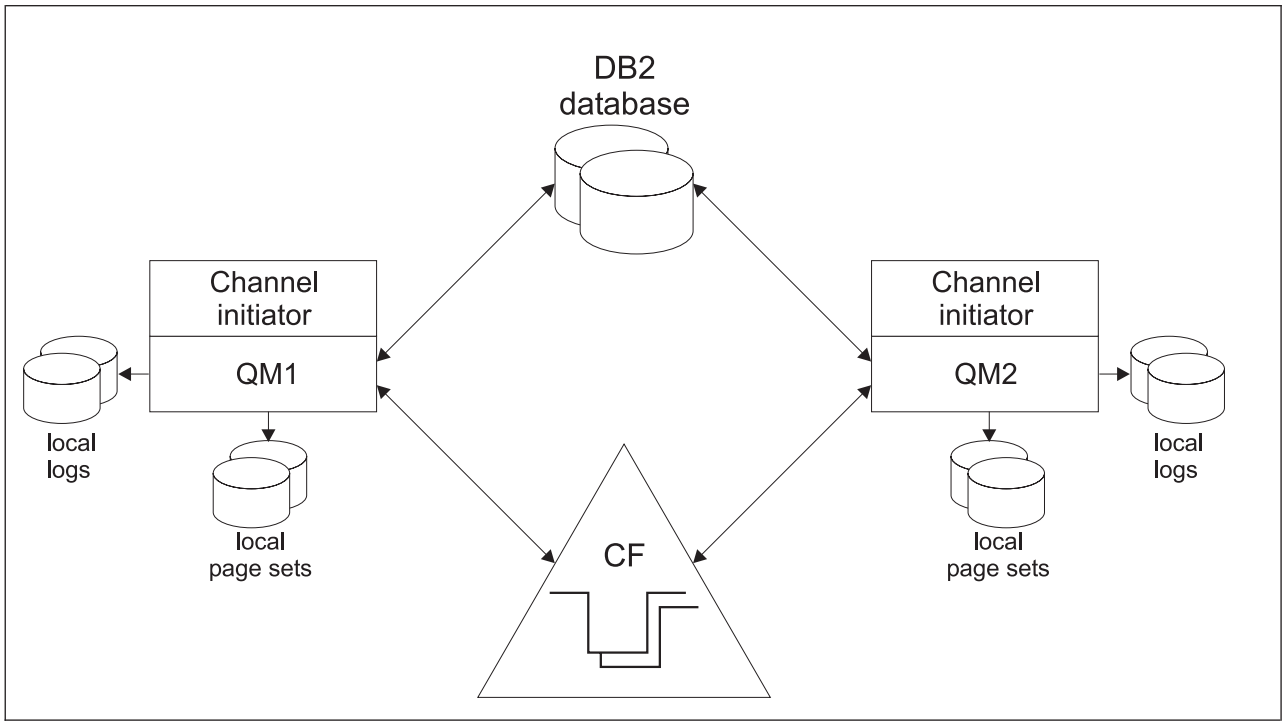


Figure 4. The components of queue managers in a queue-sharing group

When a queue manager has joined a queue-sharing group, it has access to the shared objects defined for that group, and you can use that queue manager to define new shared objects within the group. If shared queues are defined within the group, you can use this queue manager to put messages to and get messages from those shared queues. Any queue manager in the group can retrieve the messages held on a shared queue.

You can enter an MQSC command once, and have it executed on all queue managers within the queue-sharing group as if it had been entered at each queue manager individually. The *command scope* attribute is used for this. This attribute is described in "Directing commands to different queue managers" on page 71.

When a queue manager runs as a member of a queue-sharing group it must be possible to distinguish between WebSphere MQ objects defined privately to that queue manager and WebSphere MQ objects defined globally that are available to all queue managers in the queue-sharing group. The *queue-sharing group disposition* attribute is used for this. This attribute is described in "Private and global definitions" on page 70.

You can define a single set of security profiles that control access to WebSphere MQ objects anywhere within the group. This means that the number of profiles you have to define is greatly reduced.

A queue manager can belong to only one queue-sharing group, and all queue managers in the group must be in the same sysplex. You specify which queue-sharing group the queue manager belongs to in the system parameters at startup.

Where are shared queue messages held?

Messages that are put onto shared queues are not stored on page sets and do not use buffer pools.

The messages in shared queues have entries on list structures in the zSeries® Coupling Facility. Many queue managers in the same sysplex can access those messages.

Shared queue messages of size less than or equal to 63KB are held entirely in the CF list structure defined for that queue.

Shared queue messages of size greater than 63KB have their message data held as one or more binary large objects (BLOBs) in a DB2 table which is shared by a DB2 data sharing group.

Messages put on a shared queue are stored in a Coupling Facility structure until they are retrieved by an MQGET. Coupling Facility operations are used to:

- search for the next retrievable message
- lock uncommitted messages on shared queues
- notify interested queue managers about the arrival of committed messages

MQPUT and MQGET operations on persistent messages are recorded on the log of the queue manager performing that operation. This minimizes the risk of data loss in the event of a Coupling Facility failure.

The Coupling Facility

The messages held on shared queues are actually stored inside a Coupling Facility. The Coupling Facility lies outside any of the z/OS images in the sysplex and is typically configured to run on a different power supply. The Coupling Facility is therefore resilient to software failures and you can configure it so that it is resilient to hardware failures or power-outages. This means that messages stored in the Coupling Facility are highly available.

Each Coupling Facility list structure used by WebSphere MQ is dedicated to a specific queue-sharing group, but a Coupling Facility can hold structures for more than one queue-sharing group. Queue managers in different queue-sharing groups cannot share data. Up to 32 queue managers in a queue-sharing group can connect to a Coupling Facility list structure at the same time.

A single Coupling Facility list structure can contain up to 512 shared queues. The amount of message data is limited by the size of the list structure. The size of the list structure is restricted by the following factors:

- It must lie within a single Coupling Facility.

- It might share the available Coupling Facility storage with other structures for WebSphere MQ and other products.

The CF structure object

The queue manager's use of a Coupling Facility structure is specified in a CF structure (CFSTRUCT) WebSphere MQ object.

These structure objects are stored in DB2.

When using z/OS commands or definitions relating to a Coupling Facility structure, the first four characters of the name of the queue-sharing group are required. However, a WebSphere MQ CFSTRUCT object always exists within a single queue-sharing group, and so its name does not include the first four characters of the name of the queue-sharing group. For example, CFSTRUCT(MYDATA) defined in queue-sharing group starting with SQ03 would use Coupling Facility list structure SQ03MYDATA.

CF structures have a CFLEVEL attribute that determines their functional capability:

- 1, 2 - can be used for nonpersistent messages less than 63KB
- 3 - can be used for persistent and nonpersistent messages less than 63KB
- 4 - can be used for persistent and nonpersistent messages up to 100MB

Backup and recovery

You can back up Coupling Facility list structures using the WebSphere MQ command BACKUP CFSTRUCT. This puts a copy of the persistent messages currently within the CF structure onto the active log data set of the queue manager making the backup, and writes a record of the backup to DB2.

In the event of a Coupling Facility failure, you can use the WebSphere MQ command RECOVER CFSTRUCT. This uses the backup record from DB2 to locate and restore persistent messages from the backup of the CF structure. Any activity since the last backup is replayed using the logs of all the queue managers in the queue-sharing group, and the CF structure is then restored up to the point before the failure.

For more information about the BACKUP CFSTRUCT and RECOVER CFSTRUCT commands, see the WebSphere MQ Script (MQSC) Command Reference.

Advantages of using shared queues

The shared queue architecture, where cloned servers pull work from a single shared queue, has some very useful properties:

- It is scalable, by adding new instances of the server application, or even adding a new z/OS image with a queue manager (in the queue-sharing group) and a copy of the application.
- It is highly available.
- It naturally performs *pull* workload balancing, based on the available processing capacity of each queue manager in the queue-sharing group.

High availability

The following examples illustrate how you can use a shared queue to increase application availability.

Consider a WebSphere MQ scenario where client applications running in the network want to make requests of server applications running on z/OS. The client application constructs a request message and places it on a request queue. The client then waits for a reply from the server, sent to the reply-to queue named in the message descriptor of the request message.

WebSphere MQ manages the transportation of the request message from the client machine to the server's input queue on z/OS and of the server's response back to the client. By defining the server's input queue as a shared queue, any messages put to the queue can be retrieved on any queue manager in the queue-sharing group. This means that you can configure a queue manager on each z/OS image in the sysplex and, by connecting them all to the same queue-sharing group, any one of them can access messages on the server's input queue.

Messages on the server's input queue are still available, even if one of the queue managers terminates abnormally or you have to stop it for administrative reasons. You can take an entire z/OS image off-line and the messages will still be available.

To take advantage of this availability of messages on a shared queue, run an instance of the server application on each z/OS image in the sysplex to provide higher server application capacity and availability, as shown in Figure 5 on page 15.

One instance of the server application retrieves a request message from the shared queue and, based on the content, performs its processing, producing a result that is sent back to the client as a WebSphere MQ message. The response message is destined for the reply-to queue and reply-to queue manager named in the message descriptor of the request message.

There are a number of options that you can use to configure the return path; these are discussed in "Distributed queuing and queue-sharing groups" on page 15.

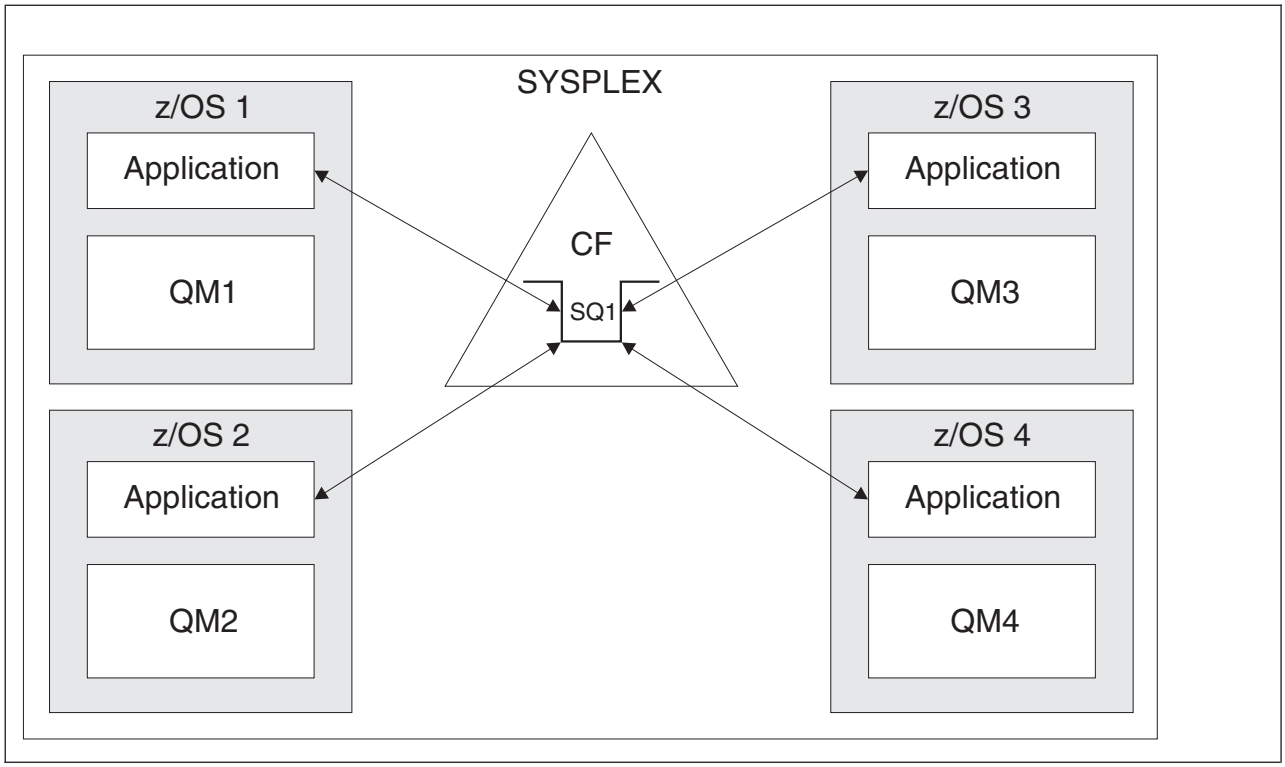


Figure 5. Multiple instances of an application servicing a shared queue

Peer recovery:

To further enhance the availability of messages in a queue-sharing group, WebSphere MQ detects if another queue manager in the group disconnects from the Coupling Facility abnormally and completes units of work for that queue manager that are still pending, where possible. This is known as *peer recovery.*

Suppose a queue manager terminates abnormally at a point where an application has retrieved a request message from a queue in syncpoint, but has not yet put the response message or committed the unit of work. Another queue manager in the queue-sharing group detects the failure, and backs out the in-flight units of work being performed on the failed queue manager. This means that the request message is put back on to the request queue and is available for one of the other server instances to process, without waiting for the failed queue manager to restart.

If WebSphere MQ cannot resolve a unit of work automatically, you can resolve the shared portion manually to enable another queue manager in the queue-sharing group to continue processing that work.

Distributed queuing and queue-sharing groups

To complement the high availability of messages on shared queues, the distributed queuing component of WebSphere MQ has additional functions to provide the following:

- Higher availability to the network.
- Increased capacity for inbound network connections to the queue-sharing group.

Figure 6 illustrates distributed queuing and queue-sharing groups. It shows two queue managers within a sysplex, both of which belong to the same queue-sharing group. They can both access shared queue SQ1. Queue managers in the network (on AIX® and Windows® for example) can put messages onto this queue through the channel initiator of either queue manager. Cloned applications on both queue managers service the queue.

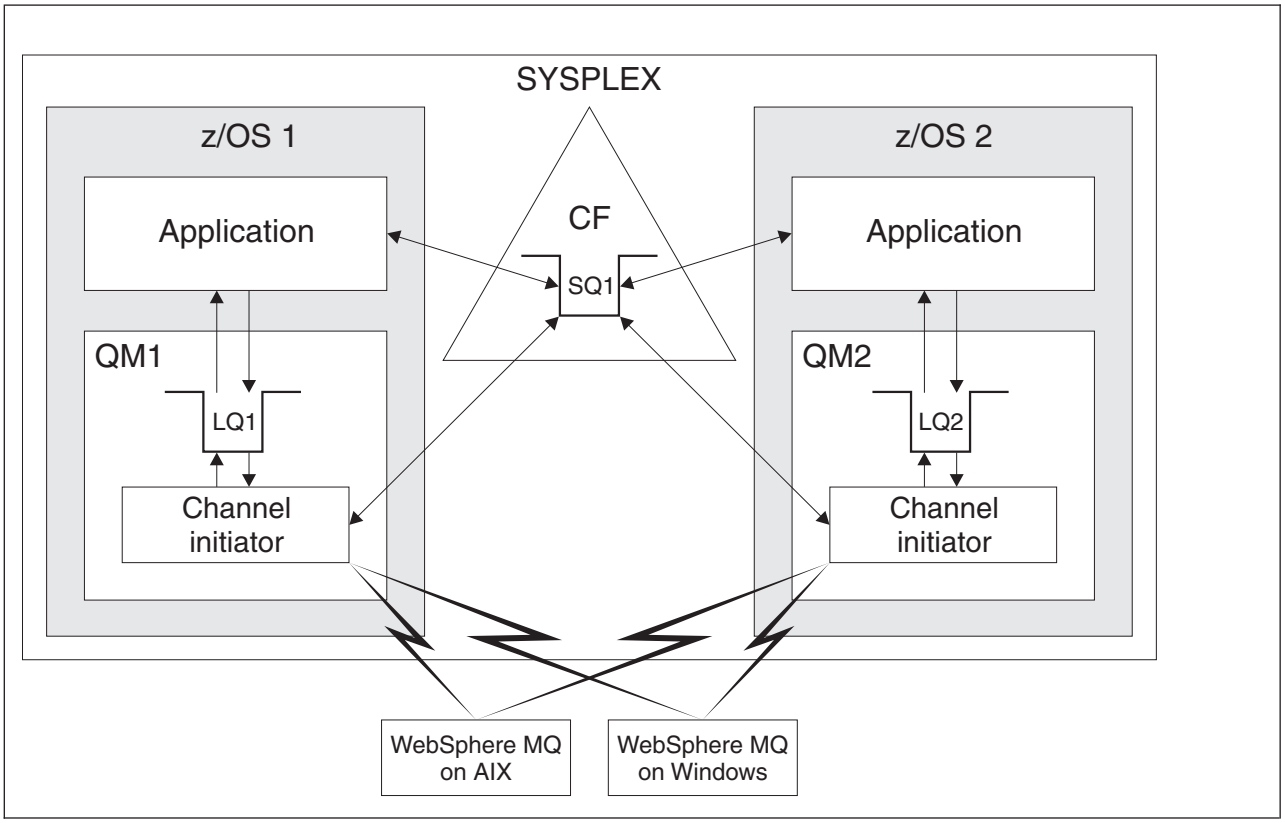


Figure 6. Distributed queuing and queue-sharing groups

Shared channels

A number of networking products provide a mechanism to hide server failures from the network, or to balance inbound network requests across a set of eligible servers. These include:

- VTAM generic resources
- TCP/IP Domain Name System (DNS)
- SYSPLEX Distributor

The channel initiator takes advantage of these products to exploit the capabilities of shared queues.

Shared inbound channels:

Each channel initiator in the queue-sharing group starts an additional listener task to listen on a *generic port*. This generic port is made available to the network through one of the technologies mentioned above. This means that an inbound network attach request for the generic port can be dispatched to any one of the listeners in the queue-sharing group that are listening on the generic port.

You can start a channel only on the channel initiator to which the inbound attach is directed if the channel initiator has access to a channel definition for a channel with that name. You can define a channel definition to be private to a queue manager or stored on the shared repository and available anywhere (a global definition). This means that you can make a channel definition available on any channel initiator in the queue-sharing group by defining it as a global definition.

There is an additional difference when starting a channel through the generic port; channel synchronization is with the queue-sharing group and not with an individual queue manager. For example, consider a client starting a channel through the generic port. When the channel first starts, it might start on queue manager QM1 and messages flow. If the channel stops and is restarted on queue manager QM2, information about the number of messages that have flowed is still correct because the synchronization is with the queue-sharing group.

You can use an inbound channel started through the generic port to put messages to any queue. The client does not know whether the target queue is shared or not. If the target queue is a shared queue, the client connects through any available channel initiator in a load-balanced fashion and the messages are put to the shared queue. If the target queue is not a shared queue, the messages might be put on any queue in the queue-sharing group with that name (the environment is one of replicated local queues), and the name of the queue determines the function, regardless of the hosting queue manager.

Shared outbound channels:

An outbound channel is considered to be a shared channel if it is taking messages from a shared transmission queue. If it is shared, it holds synchronization information at queue-sharing group level. This means that the channel can be restarted on a different queue manager and channel initiator instance within the queue-sharing group if the communications subsystem, channel initiator, or queue manager fails. Restarting failed channels in this way is a feature of shared channels called *peer channel recovery*.

Workload balancing:

An outbound shared channel is eligible for starting on any channel initiator within the queue-sharing group, provided that you have not specified that you want it to be started on a particular channel initiator. The channel initiator selected by WebSphere MQ is determined using the following criteria:

- Is the communications subsystem required currently available to the channel initiator?
- Is a DB2 connection available to the channel initiator?
- Which channel initiator has the lowest current workload? The workload includes channels that are active and retrying.

Shared channel summary:

Shared channels differ from private channels in the following ways:

Private channel
Tied to a single channel initiator.

- Outbound channel uses a local transmission queue.
- Inbound channel started through a local port.
- Synchronization information held in SYSTEM.CHANNEL.SYNCQ queue.

Shared Channel

Workload balanced with high availability.

- Outbound channel uses a shared transmission queue.
- Inbound channel started through a generic port.
- Synchronization information held in SYSTEM.QSG.CHANNEL.SYNCQ queue.

You specify whether a channel is private or shared when you start the channel. A shared channel can be started by triggering in the same way as a private channel. However, when a shared channel is started, WebSphere MQ performs workload balancing and starts the channel on the most appropriate channel initiator within the queue-sharing group. (If required, you can specify that a shared channel is to be started on a particular channel initiator.)

Shared channel status:

The channel initiators in a queue-sharing group maintain a shared channel-status table in DB2. This records which channels are active on which channel initiators. The shared channel-status table is used if there is a channel initiator or communications system failure. It indicates which channels need to be restarted on a different channel initiator in the queue-sharing group.

Intra-group queuing

Intra-group queueing (IGQ) now supports large messages, the largest being 100 MB *minus* the length of the transmission queue header.

You can perform fast message transfer between queue managers in a queue-sharing group without defining channels. This uses a system queue called the SYSTEM.QSG.TRANSMIT.QUEUE, which is a shared transmission queue. Each queue manager in the queue-sharing group starts a task called the intra-group queuing agent, which waits for messages to arrive on this queue that are destined for their queue manager. When such a message is detected, it is removed from the queue and placed on the correct destination queue.

Standard name resolution rules are used but, if intra-group queuing is enabled and the target queue manager is within the queue-sharing group, the SYSTEM.QSG.TRANSMIT.QUEUE is used to transfer the message to the correct destination queue manager instead of using a transmission queue and channel.

You enable intra-group queuing through a queue manager attribute. Intra-group queuing moves nonpersistent messages outside syncpoint, and persistent messages within syncpoint. If it finds a problem delivering messages to the target queue, intra-group queuing tries to put them to the dead-letter queue. If the dead-letter queue is full or undefined, nonpersistent messages are discarded, but persistent messages are backed out and returned to the SYSTEM.QSG.TRANSMIT.QUEUE, and the IGQ agent tries to deliver the messages until it is successful.

An inbound shared channel that receives a message destined for a queue on a different queue manager in the queue-sharing group can use intra-group queuing to *hop* the message to the correct destination.

There might be times when you want the local queue manager to put a message directly to the target queue if the target queue is a shared queue, rather than the message first being transferred to the target queue manager. You can use the queue manager attribute SQQMNAME to control this. If you set the value of

SQQMNAME to USE, the MQOPEN command is performed on the queue manager specified by the ObjectQMgrName. However, if the target queue is a shared queue and you set the value of SQQMNAME to IGNORE, and the ObjectQMgrName is that of another queue manager in the queue-sharing group, the shared queue is opened on the local queue manager. If the local queue manager cannot open the target queue, or put a message to the queue, the message is transferred to the specified ObjectQMgrName through either IGQ or an MQ channel.

If you use this feature, users must have the same access to the queues on each queue manager in the queue-sharing group.

Clusters and queue-sharing groups

You can make your shared queues available to a cluster in a single definition. To do this you specify the name of the cluster when you define the shared queue.

Users in the network see the shared queue as being hosted by each queue manager within the queue-sharing group (the shared queue is not advertised as being hosted by the queue-sharing group). Clients can start sessions with any members of the queue-sharing group to put messages to the same shared queue.

Figure 7 shows how members of a cluster can access a shared queue through any member of the queue-sharing group.

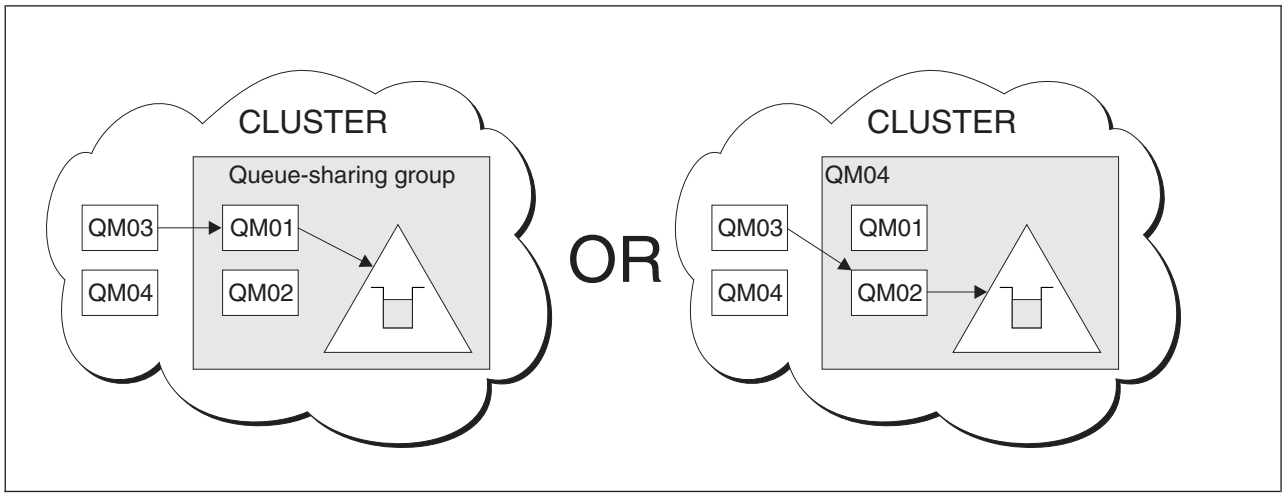


Figure 7. A queue-sharing group as part of a cluster

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 1. Where to find more information about shared queues and queue-sharing groups

Topic	Where to look
Queue-sharing group recovery	"Recovery and restart" on page 44
Queue-sharing group security	"Security" on page 59

Table 1. Where to find more information about shared queues and queue-sharing groups (continued)

Topic	Where to look
Private and global object definitions Directing commands to different queue managers	"Commands" on page 69
Planning your Coupling Facility environment	"Defining Coupling Facility resources" on page 113
Planning your DB2 environment	"Planning your DB2 environment" on page 118
Setting up your shared queues System parameters	WebSphere MQ for z/OS System Setup Guide
Utility programs Migrating queues	WebSphere MQ for z/OS System Administration Guide
Console messages	WebSphere MQ for z/OS Messages and Codes
MQSC commands	WebSphere MQ Script (MQSC) Command Reference
WebSphere MQ clusters	WebSphere MQ Queue Manager Clusters
WebSphere MQ distributed queuing Channel name resolution	WebSphere MQ Intercommunication
Writing applications	WebSphere MQ Application Programming Guide
MQCONNX call	WebSphere MQ Application Programming Reference

Storage management

This chapter discusses how WebSphere MQ for z/OS manages storage. It contains the following sections:

- "Page sets"
- "Storage classes" on page 22
- "Buffers and buffer pools" on page 23
- "Where to find more information" on page 25

Page sets

A *page set* is a VSAM linear data set that has been specially formatted to be used by WebSphere MQ. Page sets are used to store most messages and object definitions.

The exceptions to this are global definitions, which are stored in a shared repository on DB2, and the messages on shared queues. These are not stored on

the queue manager page sets. Shared queues are discussed in “Shared queues and queue-sharing groups” on page 9, and global definitions are discussed in “Private and global definitions” on page 70.

WebSphere MQ page sets can be up to 64 GB in size. Each page set is identified by a page set identifier (PSID), an integer in the range 00 through 99. Each queue manager must have its own page sets.

WebSphere MQ uses page set zero (PSID=00) to store object definitions and other important information relevant to the queue manager. For normal operation of WebSphere MQ it is essential that page set zero does not become full, so do not use it to store messages.

To improve the performance of your system, you should also separate short-lived messages from long-lived messages by placing them on different page sets.

You must format page sets, and WebSphere MQ provides a FORMAT utility for this. This is described in the WebSphere MQ for z/OS System Administration Guide. Page sets must also be defined to the WebSphere MQ subsystem, this is described in the WebSphere MQ for z/OS System Setup Guide.

WebSphere MQ for z/OS can be configured to expand a page set dynamically if it becomes full. WebSphere MQ continues to expand the page set if required until 119 logical extents exist, provided that there is sufficient disk storage space available. The extents can span volumes if the linear data set is defined in this way, however, WebSphere MQ cannot expand the page sets beyond 64 GB.

You cannot use page sets from one WebSphere MQ queue manager on a different WebSphere MQ queue manager, or change the queue manager name. If you want to transfer the data from one queue manager to another, you must unload all the objects and messages from the first queue manager and reload them onto another.

In previous releases, page sets were limited to 4 GB. It is not possible to modify such existing page sets to be larger than 4 GB. Instead, you must create new page sets with extended addressability and extended format attributes.

It is not possible to use page sets greater than 4 GB in a queue manager running a release prior to V6. During the migration period, when it is likely that you might need to fall back to a previous release of code:

- Do not change page set 0 to be greater than 4 GB.
- Other page sets greater than 4 GB will be left offline when restarting a queue manager with a previous release.

A procedure for migrating existing page sets to be capable of expanding beyond 4 GB is documented in the *WebSphere MQ for z/OS System Administration Guide* (see Chapter 10 ’Managing page sets’, heading ’Defining a page set to be larger than 4 GB’).

It is possible for an administrator to dynamically add page sets to a running queue manager, or remove page sets from a running queue manager (with the exception of page set zero). The DEFINE PSID command can run after the queue manager restart has completed, only if the command contains the DSN keyword.

Storage classes

A *storage class* maps one or more queues to a page set. This means that messages for that queue are stored on that page set.

Storage classes allow you to control where non-shared message data is stored for administrative, data set space and load management, or application isolation purposes. You can also use storage classes to define the XCF group and member name of an IMS region if you are using the IMS bridge (described in "WebSphere MQ and IMS" on page 93).

Shared queues do not use storage classes to obtain a page set mapping because the messages on them are not stored on page sets.

How storage classes work

- You define a storage class, using the DEFINE STGCLASS command, specifying a page set identifier (PSID).
- When you define a queue, you specify the storage class in the STGCLASS attribute.

In the following example, the local queue QE5 is mapped to page set 21 through storage class ARC2.

```
DEFINE STGCLASS(ARC2) PSID(21)
DEFINE QLOCAL(QE5) STGCLASS(ARC2)
```

This means that messages that are put on the queue QE5 are stored on page set 21 (if they stay on the queue long enough to be written to DASD).

More than one queue can use the same storage class, and you can define as many storage classes as you like. For example, you can extend the previous example to include more storage class and queue definitions, as follows:

```
DEFINE STGCLASS(ARC1) PSID(05)
DEFINE STGCLASS(ARC2) PSID(21)
DEFINE STGCLASS(MAXI) PSID(05)
DEFINE QLOCAL(QE1) STGCLASS(ARC1) ...
DEFINE QLOCAL(QE2) STGCLASS(ARC1) ...
DEFINE QLOCAL(QE3) STGCLASS(MAXI) ...
DEFINE QLOCAL(QE4) STGCLASS(ARC2) ...
DEFINE QLOCAL(QE5) STGCLASS(ARC2) ...
```

In Figure 8 on page 23, both storage classes ARC1 and MAXI are associated with page set 05. Therefore, the queues QE1, QE2, and QE3 are mapped to page set 05. Similarly, storage class ARC2 associates queues QE4 and QE5 with page set 21.

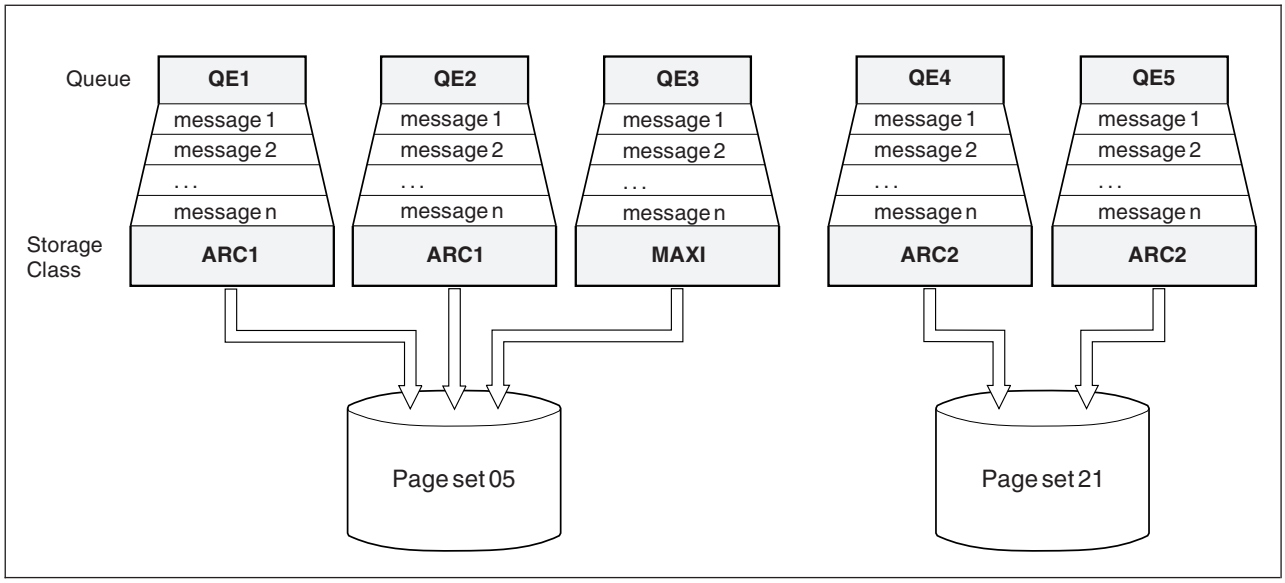


Figure 8. Mapping queues to page sets through storage classes

If you define a queue without specifying a storage class, WebSphere MQ uses a default storage class.

If a message is put on a queue that names a nonexistent storage class, the application receives an error. You must alter the queue definition to give it an existing storage class name, or create the storage class named by the queue.

You can change a storage class only when:

- All queues that use this storage class are empty, and have no uncommitted activity.
- All queues that use this storage class are closed.

Buffers and buffer pools

For efficiency, WebSphere MQ uses a form of caching whereby messages (and object definitions) are stored temporarily in buffers before being stored in page sets on DASD. Short-lived messages, that is, messages that are retrieved from a queue shortly after they are received, might only ever be stored in the buffers. However, this is all transparent to the user because the buffers are controlled by a buffer manager, which is a component of WebSphere MQ.

The buffers are organized into *buffer pools*. You can define up to 16 buffer pools (0 through 15) for each queue manager; you are recommended to use the minimal number of buffer pools consistent with the object and message type segregation outlined in Figure 9 on page 24, and any data isolation requirements your application might have. Each buffer is 4 KB long. The maximum number of buffers is determined by the amount of storage available in the queue manager address space; do not use more than about 70% of the space for buffers. Usually, the more buffers you have, the more efficient the buffering and the better the performance of WebSphere MQ.

Figure 9 on page 24 shows the relationship between messages, buffers, buffer pools, and page sets. A buffer pool is associated with one or more page sets; each page set is associated with a single buffer pool.

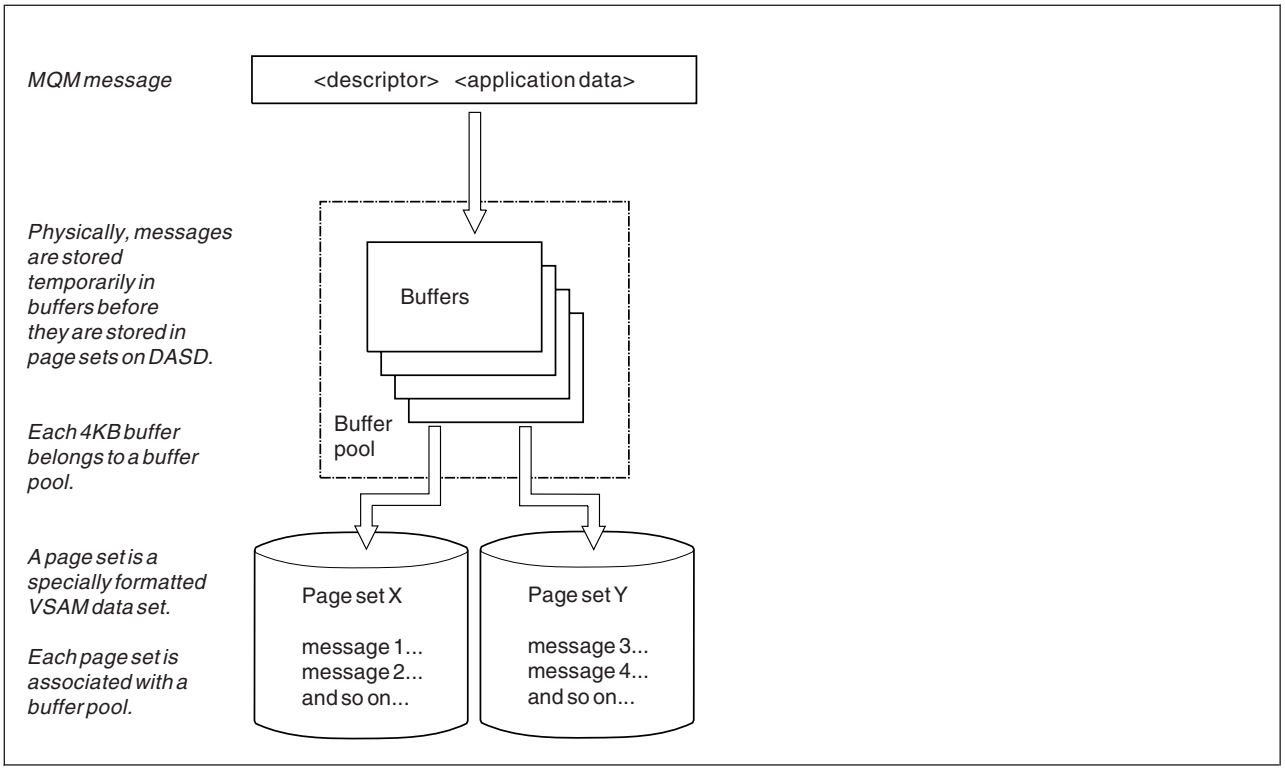


Figure 9. Buffers, buffer pools, and page sets

It is possible for an administrator to dynamically issue commands for modifying buffer pool size, numbers, and their association with page sets at any time (not just at queue manager restart), using the ALTER BUFFPOOL command.

If a buffer pool is too small, WebSphere MQ issues message CSQP020E. You can then dynamically add more buffers to the affected buffer pool (note that you may have to remove buffers from other buffer pools to do this).

You specify the number of buffers in a pool with the DEFINE BUFFPOOL command, and you dynamically resize buffer pools with the ALTER BUFFPOOL command. You determine the current number of buffers in a pool dynamically by displaying a page set that uses the buffer pool, using the DISPLAY USAGE command. These commands are described in the WebSphere MQ Script (MQSC) Command Reference manual.

For performance reasons, do not put messages and object definitions in the same buffer pool. Use one buffer pool (say number zero) exclusively for page set zero, where the object definitions are kept. Similarly, keep short-lived messages and long-lived messages in different buffer pools and therefore on different page sets, and in different queues.

The DEFINE BUFFPOOL command cannot be used after restart to create a new buffer pool. Instead, if a DEFINE PSID command uses the DSN keyword, it may explicitly identify a buffer pool that is not currently defined. That new buffer pool will then be created.

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 2. Where to find more information about storage management

Topic	Where to look
How much storage you need	"Planning your storage and performance requirements" on page 101
How large to make your page sets and buffer pools	"Planning your page sets and buffer pools" on page 106
Defining page sets	WebSphere MQ for z/OS System Setup Guide
Formatting page sets Utility programs	WebSphere MQ for z/OS System Administration Guide
Console messages	WebSphere MQ for z/OS Messages and Codes
MQSC commands	WebSphere MQ Script (MQSC) Command Reference

Logging

WebSphere MQ maintains *logs* of data changes and significant events as they occur. The *bootstrap data set* (BSDS) stores information about the data sets that contain the logs.

This chapter contains the following sections:

- "What logs are"
- "How the log is structured" on page 29
- "How the logs are written" on page 29
- "What the bootstrap data set is for" on page 32
- "Where to find more information" on page 34

The log does not contain information for statistics, traces, or performance evaluation. The statistical and monitoring information that WebSphere MQ collects is discussed in "Monitoring and statistics" on page 79.

What logs are

Logs contain information needed for recovery. Active logs can be archived so that you can keep log data for a long period.

WebSphere MQ records all significant events as they occur in an *active log*. The log contains the information needed to recover:

- Persistent messages
- WebSphere MQ objects, such as queues
- The WebSphere MQ queue manager

The active log comprises a collection of data sets (up to 31) which are used cyclically.

You can enable log archiving so that when an active log fills a copy is made in an archive data set. Using archiving allows you to keep log data for an extended period. If you do not use archiving, the logs wrap and earlier data is overwritten. To recover a page set, or recover data in a CF structure, you need log data from when the backup of the page set or structure was taken. An archive log can be created on disk or on tape.

Archiving

Because the active log has a fixed size, WebSphere MQ copies the contents of each log data set periodically to an *archive log*, which is normally a data set on a direct access storage device (DASD) or a magnetic tape. If there is a subsystem or transaction failure, WebSphere MQ uses the active log and, if necessary, the archive log for recovery.

The archive log can contain up to 1000 sequential data sets. You can catalog each data set using the z/OS integrated catalog facility (ICF).

Archiving is an essential component of WebSphere MQ recovery. If a unit of recovery is a long-running one, log records within that unit of recovery might be found in the archive log. In this case, recovery requires data from the archive log. However, if archiving is switched off, the active log with new log records wraps, overwriting earlier log records. This means that WebSphere MQ might not be able to back out the unit of recovery and messages might be lost. The queue manager then terminates abnormally.

Therefore, in a production environment, **never switch archiving off**. If you do, you run the risk of losing data after a system or transaction failure. Only if you are running in a test environment can you consider switching archiving off. If you need to do this, use the CSQ6LOGP macro, which is described in the WebSphere MQ for z/OS System Setup Guide.

To help prevent problems with unplanned long-running units of work, WebSphere MQ issues a message (CSQJ160I or CSQJ161I) if a long-running unit of work is detected during active log offload.

Dual logging

In dual logging, each log record is written to two different active log data sets in order to minimize the likelihood of data loss problems during restart.

You can configure WebSphere MQ to run with either *single logging* or *dual logging*. With single logging, log records are written once to an active log data set. Each active log data set is a single-extent VSAM linear data set (LDS). With dual logging, each log record is written to two different active log data sets. Dual logging minimizes the likelihood of data loss problems during restart.

See the related link for a discussion of single and dual logging.

Log shunting

Log shunting causes the log records for some units of work to be written further down the log. This reduces the amount of log data that must be read at queue manager restart, or backout, for long running or long term in-doubt units of work.

When a unit of work is considered to be long, a representation of each log record is written further down the log. This technique is known as *shunting*. When the whole of the unit of work has been processed, the unit of work is in a *shunted* state. Any backout or restart activity relating to the shunted unit of work can use the shunted log records instead of using the original unit of work log records.

Detecting a long-running unit of work is a function of the checkpoint process. At checkpoint time, each active unit of work is checked to establish whether it needs to be shunted. If the unit of work has been through two prior checkpoints since it was created, or since it was last shunted, the unit of work is suitable to be shunted. This means that a single unit of work might be shunted more than once. This is known as a *multi-shunted* unit of work.

Log shunting is always active, and runs whether or not log archiving is enabled.

Note: Although all log records for a unit of work are shunted, the entire content of each record is not shunted, only the part that is necessary for backout. This means that the amount of log data written is kept to a minimum, and that shunted records cannot be used if a page set failure occurs. A long running unit of work is one that has been running for more than three queue manager checkpoints.

For more information about log shunting, see the WebSphere MQ for z/OS System Administration Guide, Chapter 8 'Managing the logs'.

Log data

The log can contain up to 140 million million ($1.4*10^{14}$) bytes. Each byte can be addressed by its offset from the beginning of the log, and that offset is known as its *relative byte address* (RBA).

The RBA is referenced by a 6-byte field giving a total addressable range of 2^{48} bytes. However, when WebSphere MQ detects that the used range is beyond x'700 000 000 000' (that is, more than $1.2*10^{14}$ bytes of the log have been used), messages CSQI045, CSQI046 and CSQI047 are issued, warning you to reset the log RBA. Do not allow the log RBA to go beyond 2^{47} bytes (x'800 000 000 000').

The log is made up of *log records*, each of which is a set of log data treated as a single unit. A log record is identified either by the RBA of the first byte of its header, or by its log record sequence number (LRSN). The RBA or LRSN uniquely identifies a record that starts at a particular point in the log.

Whether you use the RBA or LRSN to identify log points depends on whether you are using queue-sharing groups. In a queue-sharing environment, you cannot use the relative byte address to uniquely identify a log point, because multiple queue managers can update the same queue at the same time, and each has its own log. To solve this, the log record sequence number is derived from a timestamp value, and does not necessarily represent the physical displacement of the log record within the log.

Each log record has a header that gives its type, the WebSphere MQ subcomponent that made the record, and, for unit of recovery records, a unit of recovery identifier.

There are four types of log record, described under the following headings:

- "Unit-of-recovery log records" on page 28
- "Checkpoint records" on page 28

- "Page set control records"
- "CF structure backup records"

Unit-of-recovery log records

Most of the log records describe changes to WebSphere MQ queues. All such changes are made within units of recovery.

WebSphere MQ uses special logging techniques involving *undo/redo* and *compensating log records* to reduce restart times and improve system availability.

One effect of this is that the restart time is bounded. If a failure occurs during a restart so that the queue manager has to be restarted a second time, all the recovery activity that completed to the point of failure in the first restart does not need to be reapplied during a second restart. This means that successive restarts do not take progressively longer times to complete.

Checkpoint records

To reduce restart time, WebSphere MQ takes periodic checkpoints during normal operation. These occur as follows:

- When a predefined number of log records has been written. This number is defined by the checkpoint frequency operand called LOGLOAD of the system parameter macro CSQ6SYSP, described in the WebSphere MQ for z/OS System Setup Guide.
- At the end of a successful restart.
- At normal termination.
- Whenever WebSphere MQ switches to the next active log data set in the cycle.

At the time a checkpoint is taken, WebSphere MQ issues the DISPLAY CONN command (described in the WebSphere MQ Script (MQSC) Command Reference) internally so that a list of connections currently in doubt is written to the z/OS console log.

Page set control records

These records register the page sets known to the WebSphere MQ queue manager at each checkpoint, and record information about the log ranges required to perform media recovery of the page set at the time of the checkpoint.

Certain dynamic changes to page sets and buffer pools are also written as page set control records, so that the changes can be recovered and automatically reinstated at the next queue manager restart.

CF structure backup records

These records hold data read from a Coupling Facility list structure in response to a BACKUP CFSTRUCT command. In the unlikely event of a Coupling Facility structure failure, these records are used, together with unit of recovery records, by the RECOVER CFSTRUCT command to perform media recovery of the Coupling Facility structure to the point of failure.

How the log is structured

Each active log data set must be a VSAM linear data set (LDS). The physical output unit written to the active log data set is a 4 KB control interval (CI). Each CI contains one VSAM record.

Physical and logical log records

One VSAM CI is a *physical* record. The information logged at a particular time forms a *logical* record, whose length varies independently of the space available in the CI. So one physical record might contain:

- Several logical records
- One or more logical records and part of another logical record
- Part of one logical record only

The term "log record" refers to the *logical* record, regardless of how many *physical* records are needed to store it.

How the logs are written

WebSphere MQ writes each log record to a DASD data set called the *active log*. When the active log is full, WebSphere MQ copies its contents to a DASD or tape data set called the *archive log*. This process is called *off-loading*.

Figure 10 on page 30 illustrates the process of logging. Log records typically go through the following cycle:

1. WebSphere MQ notes changes to data and significant events in recovery log records.
2. WebSphere MQ processes recovery log records and breaks them into segments, if necessary.
3. Log records are placed sequentially in *output log buffers*, which are formatted as VSAM CIs. Each log record is identified by a relative byte address in the range zero through 2^{48}-1.
4. The CIs are written to a set of predefined DASD active log data sets, which are used sequentially and recycled.
5. If archiving is active, as each active log data set becomes full, its contents are automatically off-loaded to a new archive log data set.

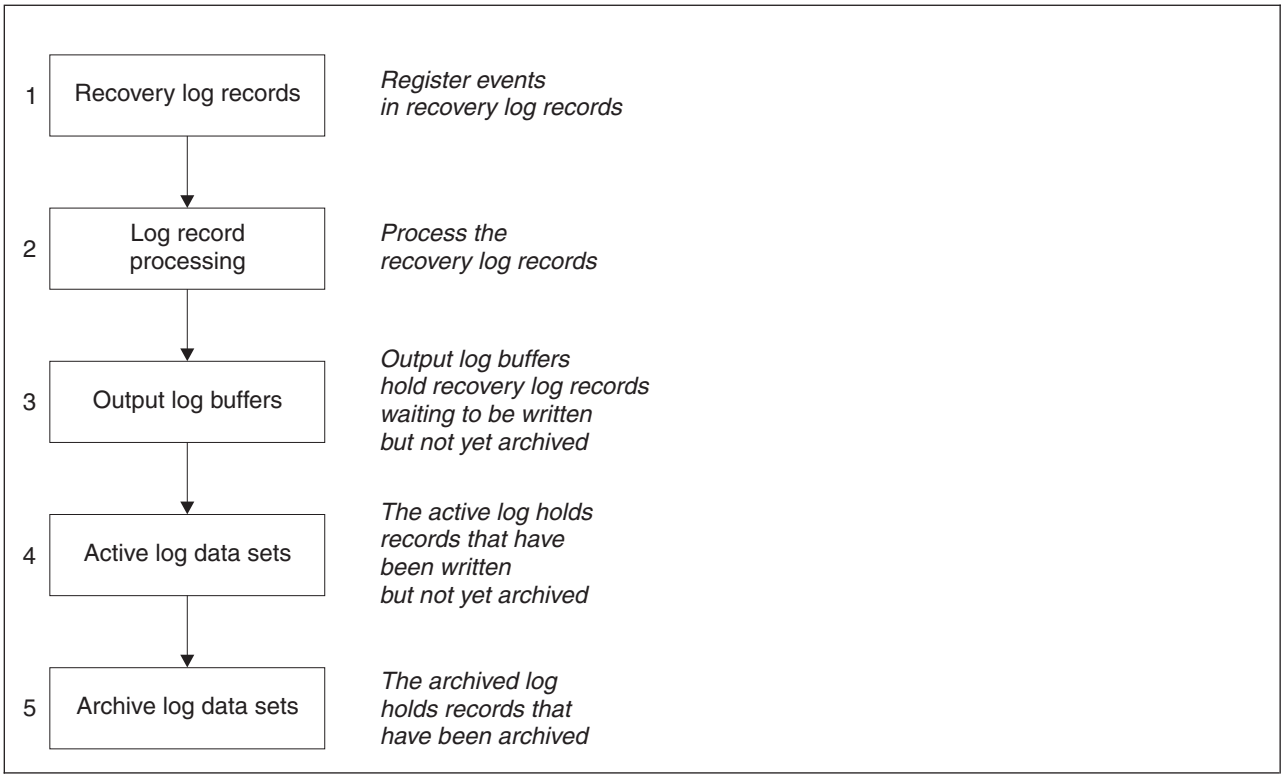


Figure 10. The logging process

When the active log is written

The in-storage log buffers are written to an active log data set whenever any of the following occur:

- The log buffers become full.
- The write threshold is reached (as specified in the CSQ6LOGP macro).
- Certain significant events occur, such as a commit point, or when a WebSphere MQ BACKUP CFSTRUCT command is issued.

When the queue manager is initialized, the active log data sets named in the BSDS are dynamically allocated for exclusive use by the queue manager and remain allocated exclusively to WebSphere MQ until the queue manager terminates.

Dynamically adding log data sets

It is possible to dynamically define new active log data sets while the queue manager is running. This feature alleviates the problem of a queue manager hang when archiving is not able to offload active logs due to a transient problem. See the DEFINE LOG command in the WebSphere MQ Script (MQSC) Command Reference for more information.

Note: To redefine or remove active logs you must terminate and restart the queue manager.

When the archive log is written

The process of copying active logs to archive logs is called *off-loading*. The relation of off-loading to other logging events is shown schematically in Figure 11.

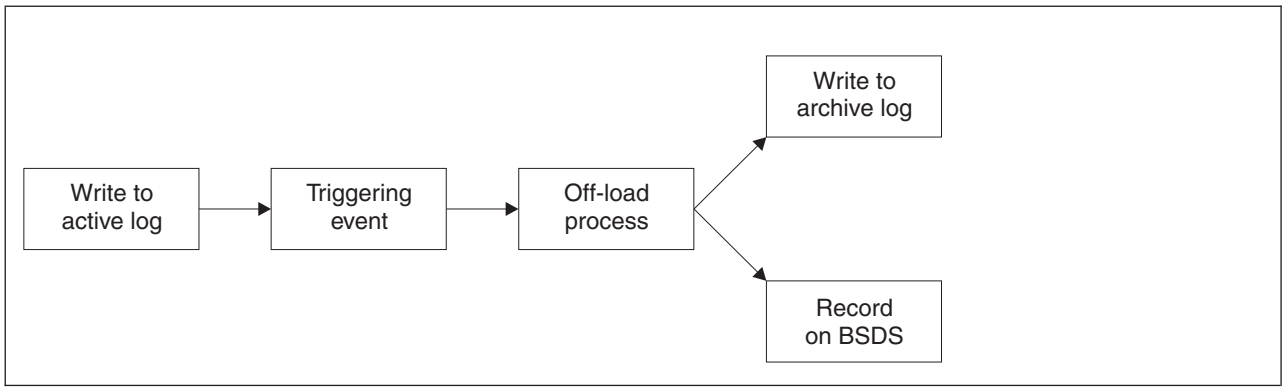


Figure 11. The off-loading process

Triggering an off-load:

The off-load of an active log to an archive log can be triggered by several events. For example:

- Filling an active log data set.
- Using the MQSC ARCHIVE LOG command.
- An error occurring while writing to an active log data set.

The data set is truncated before the point of failure, and the record that was not written becomes the first record of the new data set. Off-load is triggered for the truncated data set as for a normal full log data set. If there are dual active logs, both copies are truncated so that the two copies remain synchronized.

Message CSQJ110E is issued when the last available active log is 5% full and at 5% increments thereafter, stating the percentage of the log's capacity that is in use. If all the active logs become full, WebSphere MQ stops processing, until off-loading occurs, and issues this message:

```
CSQJ111A +CSQ1 OUT OF SPACE IN ACTIVE LOG DATA SETS
```

The off-load process:

When all the active logs become full, WebSphere MQ runs an off-load and halts processing until the off-load has been completed. If the off-load processing fails when the active logs are full, WebSphere MQ abends.

When an active log is ready to be off-loaded, a request is sent to the z/OS console operator to mount a tape or prepare a DASD unit. The value of the ARCWTOR logging option (discussed in the WebSphere MQ for z/OS System Setup Guide) determines whether the request is received. If you are using tape for off-loading, specify ARCWTOR=YES. If the value is YES, the request is preceded by a WTOR (message number CSQJ008E) telling the operator to prepare an archive log data set to be allocated.

The operator need not respond to this message immediately. However, delaying the response delays the off-load process. It does not affect WebSphere MQ performance unless the operator delays the response for so long that WebSphere MQ runs out of active logs.

The operator can respond by canceling the off-load. In this case, if the allocation is for the first copy of dual archive data sets, the off-load is merely delayed until the next active log data set becomes full. If the allocation is for the second copy, the archive process switches to single copy mode, but for this data set only.

Interruptions and errors while off-loading:

A request to stop the queue manager does not take effect until off-loading has finished. If WebSphere MQ fails while off-loading is in progress, off-load begins again when the queue manager is restarted. Off-load handling of I/O errors on the logs is discussed in the WebSphere MQ for z/OS System Administration Guide.

Messages during off-load:

Off-load messages are sent to the z/OS console by WebSphere MQ and the off-load process. You can use these messages to find the RBA ranges in the various log data sets. For an explanation of the off-load messages, see the WebSphere MQ for z/OS Messages and Codes manual.

WebSphere MQ and SMS

WebSphere MQ parameters enable you to specify Storage Management Subsystem (MVS/DFP™ SMS) storage classes when allocating WebSphere MQ archive log data sets dynamically. WebSphere MQ initiates the archiving of log data sets, but you can use SMS to perform allocation of the archive data set.

What the bootstrap data set is for

The *bootstrap data set* (BSDS) is a VSAM key-sequenced data set (KSDS) that holds information needed by WebSphere MQ. It contains the following:

- An inventory of all active and archived log data sets known to WebSphere MQ. WebSphere MQ uses this inventory to:
  - Track the active and archived log data sets
  - Locate log records so that it can satisfy log read requests during normal processing
  - Locate log records so that it can handle restart processing

  WebSphere MQ stores information in the inventory each time an archive log data set is defined or an active log data set is reused. For active logs, the inventory shows which are full and which are available for reuse. The inventory holds the relative byte address (RBA) of each portion of the log held in that data set.
- A *wrap-around* inventory of all recent WebSphere MQ activity. This is needed if you have to restart the queue manager.

The BSDS is required if the queue manager has an error and you have to restart it. WebSphere MQ **must** have a BSDS. To minimize the likelihood of problems during a restart, you can configure WebSphere MQ with dual BSDSs, each recording the same information. This is known as running in *dual mode*. If possible, the copies

should be on separate volumes. This reduces the risk of them both being lost if the volume is corrupted or destroyed. You should use dual BSDSs rather than dual write to DASD.

The BSDS is set up when WebSphere MQ is customized and you can manage the inventory using the change log inventory utility (CSQJU003). This utility is discussed in the WebSphere MQ for z/OS System Administration Guide. It is referenced by a DD statement in the queue manager startup procedure.

Normally, WebSphere MQ keeps duplicate copies of the BSDS. If an I/O error occurs, it deallocates the failing copy and continues with a single BSDS. You can restore dual mode operation, this is described in the WebSphere MQ for z/OS System Administration Guide.

The active logs are first registered in the BSDS when WebSphere MQ is installed. You cannot replace the active logs without terminating and restarting the queue manager.

Archive log data sets are allocated dynamically. When one is allocated, the data set name is registered in the BSDS. The list of archive log data sets expands as archives are added, and wraps when a user-determined number of entries has been reached. The maximum number of entries is 1000 for single archive logging and 2000 for dual logging.

You can use a tape management system to delete the archive log data sets (WebSphere MQ does not have an automated method). Therefore, the information about an archive log data set can be in the BSDS long after the archive log data set has been deleted by the system administrator.

Conversely, the maximum number of archive log data sets could have been exceeded, and the data from the BSDS dropped long before the data set has reached its expiry date.

You can use the following MQSC command to determine the extent of the log, and the name of the active or archive log data set holding the earliest log RBA, required for various types of media or queue manager recovery:

```
DISPLAY USAGE TYPE(DATASET)
```

If the system parameter module specifies that archive log data sets are cataloged when allocated, the BSDS points to the integrated catalog facility (ICF) catalog for the information needed for later allocations. Otherwise, the BSDS entries for each volume register the volume serial number and unit information that is needed for later allocations.

Archive log data sets and BSDS copies

Each time a new archive log data set is created, a copy of the BSDS is also created. If the archive log is on tape, the BSDS is the first data set on the first output volume. If the archive log is on DASD, the BSDS is a separate data set.

The data set names of the archive log and the BSDS copy are the same, except that the lowest-level qualifier of the archive log name begins with A and the BSDS copy begins with B, for example:

Archive log name
CSQ.ARCHLOG1.E00186.T2336229.*A*0000001

BSDS copy name
CSQ.ARCHLOG1.E00186.T2336229.*B*0000001

If there is a read error while copying the BSDS, the copy is not created, message CSQJ125E is issued, and the off-load to the new archive log data set continues without the BSDS copy.

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 3. Where to find more information about logging

Topic	Where to look
How many logs are required How large to make the logs	"Planning your logging environment" on page 120
Setting up your logs System parameter macros	WebSphere MQ for z/OS System Setup Guide
Day to day logging tasks Resolving problems with logs Utility programs	WebSphere MQ for z/OS System Administration Guide
Console messages	WebSphere MQ for z/OS Messages and Codes
MQSC commands	WebSphere MQ Script (MQSC) Command Reference

Defining your system

This chapter discusses the following topics:

- "Setting system parameters"
- "Defining system objects" on page 35
- "Tuning your queue manager" on page 38
- "Sample definitions supplied with WebSphere MQ" on page 40
- "Where to find more information" on page 43

Setting system parameters

In WebSphere MQ for z/OS, a system parameter module controls the logging, archiving, tracing, and connection environments that WebSphere MQ uses in its operation. The system parameters are specified by three assembler macros, as follows:

CSQ6SYSP
System parameters, including setting the connection and tracing environment.

CSQ6LOGP
Logging parameters.

CSQ6ARVP
Log archive parameters.

Default parameter modules are supplied with WebSphere MQ for z/OS. If these do not contain the values that you want to use, you can create your own parameter modules using the sample supplied with WebSphere MQ. The sample is thlqual.SCSQPROC(CSQ4ZPRM).

You can alter some system parameters while a queue manager is running. See the SET SYSTEM, SET LOG and SET ARCHIVE commands in the WebSphere MQ Script (MQSC) Command Reference.

Defining system objects

There are several objects that you need to define before you can use WebSphere MQ for z/OS. These are called the system objects and are described here. Sample definitions are supplied with WebSphere MQ to help you define these objects. These samples are described in "Sample definitions supplied with WebSphere MQ" on page 40.

Publish/subscribe objects

To use Publish/Subscribe you need to define the following objects:

- A local queue called the SYSTEM.RETAINED.PUB.QUEUE which is used to hold a copy of each retained publication in the queue manager. Each full topic name could have up to one retained publication stored on this queue. If your applications will make use of retained publications on many different topics, or if your retained publication messages are large messages, the requirements for storage for this queue should be carefully planned, including assigning it to its own page set if the storage requirements for it are large. To improve performance, you should define this queue with an index type of MSGID (as shown in the supplied sample queue definition).
- Two local queues, named SYSTEM.DURABLE.SUBSCRIBER.QUEUE and SYSTEM.DURABLE.SHARED.SUBSCRIBER.QUEUE which are used to hold a persistent copy of the durable subscriptions in the queue manager. To improve performance, you should define these queues with an index type of CORRELID (as shown in the supplied sample queue definition).

System default objects

The system default objects are used to provide default attributes when you define an object and do not specify the name of another object to base the definition on.

The names of the default system object definitions begin with the characters "SYSTEM.DEFAULT" or "SYSTEM.DEF". For example, the system default local queue is named SYSTEM.DEFAULT.LOCAL.QUEUE.

These objects define the system defaults for the attributes of these WebSphere MQ objects:

- Local queues
- Model queues
- Alias queues
- Remote queues
- Processes

- Namelists
- Channels
- Storage classes
- Authentication information

Shared queues are a special type of local queue, so when you define a shared queue, the definition is based on the SYSTEM.DEFAULT.LOCAL.QUEUE. You need to remember to supply a value for the Coupling Facility structure name because one is not specified in the default definition. Alternatively, you could define your own default shared queue definition to use as a basis for shared queues so that they all inherit the required attributes. Remember that you need to define a shared queue on one queue manager in the queue-sharing group only.

System command objects

The names of the system command objects begin with the characters SYSTEM.COMMAND. You must define these objects before you can use the WebSphere MQ operations and control panels to issue commands to a WebSphere MQ subsystem.

There are two system command objects:

1. The system-command input queue is a local queue on which commands are put before they are processed by the WebSphere MQ command processor. It must be called SYSTEM.COMMAND.INPUT, although an alias of SYSTEM.ADMIN.COMMAND.QUEUE for it can be defined, for compatibility with non-z/OS WebSphere MQ.
2. SYSTEM.COMMAND.REPLY.MODEL is a model queue that defines the system-command reply-to queue.

There are 2 extra objects for use by the WebSphere MQ Explorer:

- SYSTEM.MQEXPLORER.REPLY.MODEL queue
- SYSTEM.ADMIN.SVRCONN channel

Commands are normally sent using nonpersistent messages so both the system command objects should have the DEFPSIST(NO) attribute so that applications using them (including the supplied applications like the utility program and the operations and control panels) get nonpersistent messages by default. If you have an application that uses persistent messages for commands, set the DEFTYPE(PERMDYN) attribute for the reply-to queue, because the reply messages to such commands are persistent.

System administration objects

The names of the system administration objects begin with the characters SYSTEM.ADMIN.

There are seven system administration objects:

- The SYSTEM.ADMIN.CHANNEL.EVENT queue
- The SYSTEM.ADMIN.COMMAND.EVENT queue
- The SYSTEM.ADMIN.CONFIG.EVENT queue
- The SYSTEM.ADMIN.PERFM.EVENT queue
- The SYSTEM.ADMIN.QMGR.EVENT queue
- The SYSTEM.ADMIN.TRACE.ROUTE.QUEUE queue

- The SYSTEM.ADMIN.ACTIVITY.QUEUE queue

Channel queues

To use distributed queuing, you need to define the following objects:

- A local queue with the name SYSTEM.CHANNEL.SYNCQ, which is used to maintain sequence numbers and logical units of work identifiers (LUWID) of channels. To improve channel performance, you should define this queue with an index type of MSGID (as shown in the supplied sample queue definition).
- A local queue with the name SYSTEM.CHANNEL.INITQ, which is used for channel commands.

You cannot define these queues as shared queues.

Cluster queues

To use WebSphere MQ clusters, you need to define the following objects:

- A local queue called the SYSTEM.CLUSTER.COMMAND.QUEUE, which is used to communicate repository changes between queue managers. Messages written to this queue contain updates to the repository data to be applied to the local copy of the repository, or requests for repository data.
- A local queue called SYSTEM.CLUSTER.REPOSITORY.QUEUE, which is used to hold a persistent copy of the repository.
- A local queue called SYSTEM.CLUSTER.TRANSMIT.QUEUE, which is the transmission queue for all destinations in the cluster. For performance reasons you should define this queue with an index type of CORRELID (as shown in the sample queue definition).

These queues typically contain large numbers of messages.

You cannot define these queues as shared queues.

Queue-sharing group queues

To use shared channels and intra-group queuing, you need to define the following objects:

- A shared queue with the name SYSTEM.QSG.CHANNEL.SYNCQ, which is used to hold synchronization information for shared channels.
- A shared queue with the name SYSTEM.QSG.TRANSMIT.QUEUE, which is used as the transmission queue for intra-group queuing. If you are running in a queue-sharing group, you must define this queue, even if you are not using intra-group queuing.

Storage classes

You are recommended to define the following six storage classes. You must define four of them because they are required by WebSphere MQ. The other storage class definitions are recommended because they are used in the sample queue definitions.

DEFAULT (required)
This storage class is used for all message queues that are not performance critical and that don't fit in to any of the other storage classes. It is also the supplied default storage class if you do not specify one when defining a queue.

NODEFINE (required)
This storage class is used if the storage class specified when you define a queue is not defined.

REMOTE (required)
This storage class is used primarily for transmission queues, that is, system related queues with short-lived performance-critical messages.

SYSLNGLV
This storage class is used for long-lived, performance-critical messages.

SYSTEM (required)
This storage class is used for performance critical, system related message queues, for example the SYSTEM.CHANNEL.SYNQ and the SYSTEM.CLUSTER.* queues.

SYSVOLAT
This storage class is used for short-lived, performance-critical messages.

You can modify their attributes and add other storage class definitions as required.

Dead-letter queue

The dead-letter queue is used if the message destination is not valid. WebSphere MQ puts such messages on a local queue called the dead-letter queue. Although having a dead-letter queue is not mandatory, you should regarded it as essential, especially if you are using distributed queuing or one of the WebSphere MQ bridges.

Do **not** define the dead-letter queue as a shared queue.

If you decide to define a dead-letter queue, you must also tell the queue manager its name. To do this use the ALTER QMGR command, as shown in the sample.

Default transmission queue

The default transmission queue is used when no other suitable transmission queue is available for sending messages to another queue manager. If you define a default transmission queue, you must also define a channel to serve the queue. If you do not do this, messages that are put on to the default transmission queue are not transmitted to the remote queue manager and remain on the queue.

If you decide to define a default transmission queue, you must also tell the queue manager its name. To do this use the ALTER QMGR command.

Pending data queue

A queue defined for internal use, SYSTEM.PENDING.DATA.QUEUE, supports the use of durable subscriptions in a JMS publish/subscribe environment.

Tuning your queue manager

There are a number of ways in which you can improve the performance of your queue manager, which are controlled by queue manager attributes set by the ALTER QMGR command. This section discusses how you can do this by setting the maximum number of messages allowed on the queue manager, or by

performing 'housekeeping' on the queue manager. WebSphere MQ SupportPac™ *Capacity planning and tuning for WebSphere MQ for z/OS* (MP16) gives more information on performance and tuning.

Syncpoints

One of the roles of the queue manager is syncpoint control within an application. An application constructs a unit of work containing any number of MQPUT or MQGET calls terminated with an MQCMIT call. However, as the number of MQPUT or MQGET calls within the scope of one MQCMIT increases, the performance cost of the commit increases significantly.

You can limit the number of messages within any single syncpoint by using the MAXUMSGS queue manager attribute. You should not normally exceed 100 MQPUTs within a single MQCMIT.

Set a fairly low MAXUMSGS value, such as 100, to avoid performance problems, and to protect against looping applications. If you have a need for a larger value, change MAXUMSGS temporarily while the application that requires the larger value is run, rather than using a permanently high value. However, be aware that reducing the value of MAXUMSGS may cause problems to existing applications and queue manager processes such as clustering if they are already using a higher value.

Expired messages

Messages that have expired are discarded by the next appropriate MQGET call. However, if no such call occurs, the expired messages are not discarded, and, for some queues, particularly those where message retrieval is done by MessageId, CorrelId, or GroupId and the queue is indexed for performance, a large number of expired messages can accumulate. The queue manager can periodically scan any queue for expired messages, which are then deleted. You can choose how often this scanning takes place, if at all. There are two ways of doing this:

Explicit request
: You can control which queues are scanned and when. Issue the REFRESH QMGR TYPE(EXPIRY) command, specifying the queue or queues that you want to be scanned.

Periodic scan
: You can specify an expiry interval in the queue manager object by using the EXPRYINT attribute. The queue manager maintains information about the expired messages on each queue, and knows at what time a scan for expired messages is worthwhile. Each time that the EXPRYINT interval is reached, the queue manager looks for candidate queues that are worth scanning for expired messages, and scans only those queues that it deems to be worthwhile. It does not scan all queues. This avoids any CPU time being wasted on unnecessary scans.

  Shared queues are only scanned by one queue manager in the queue-sharing group. Generally, the first queue manager to restart or the first to have the EXPRYINT set performs the scan.

  Note: You must set the same EXPRYINT value for all queue managers within a queue-sharing group.

Sample definitions supplied with WebSphere MQ

The following sample definitions are supplied with WebSphere MQ in the thlqual.SCSQPROC library. You can use them to define the system objects and to customize your own objects. You can include some of them in the initialization input data sets (described in "Initialization commands" on page 76).

Table 4. WebSphere MQ sample definitions for system objects

Initialization input data set	Sample name
CSQINP1	CSQ4INP1 CSQ4INPR
CSQINP2	CSQ4INYS[1] CSQ4INSX CSQ4INSG CSQ4INSR CSQ4INSS CSQ4INSJ CSQ4INYG CSQ4INYR CSQ4INYC CSQ4INYD
Other	CSQ4DISP CSQ4INPX CSQ4IVPQ CSQ4IVPG
Note: 1. The order of these sample definitions is important: an error occurs if INYS, INSX, and INSG are ordered incorrectly.	

The CSQINP1 samples

Use the sample CSQINP1 data set thlqual.SCSQPROC(CSQ4INP1) when you are using one page set for each class of message, or thlqual.SCSQPROC(CSQ4INPR) when using multiple page sets for the major classes of message. It contains definitions of buffer pools, page set to buffer pool associations, and an ALTER SECURITY command. Include the sample in the CSQINP1 concatenation of your queue manager started task procedure.

CSQ4INSG system object sample

The sample CSQINP2 data set thlqual.SCSQPROC(CSQ4INSG) contains definitions for the following system objects for general use:

- System default objects
- System command objects
- System administration objects
- Other objects for system use

You must define the objects in this sample, but you need to do it only once when the subsystem is first started. Including the definitions in the CSQINP2 data set is

the best way to do this. They are maintained across queue manager shutdown and restart. You must not change the object names, but you can change their attributes if required.

CSQ4INSS system object sample

You can define additional system objects if you are using queue-sharing groups.

Sample data set thlqual.SCSQPROC(CSQ4INSS) contains sample commands for use with CF structures and a set of definitions for the system objects required for shared channels and intra-group queuing.

You cannot use this sample as is; you must customize it before use. Then you can include this member in the CSQINP2 DD concatenation of the queue manager startup procedure, or you can use it as input to the COMMAND function of the CSQUTIL utility to issue the required commands.

When you are defining group or shared objects, you need to include them in the CSQINP2 DD concatenation for only one queue manager in the queue-sharing group.

CSQ4INSX system object sample

You must define additional system objects if you are using distributed queuing and clustering.

Sample data set thlqual.SCSQPROC(CSQ4INSX) contains the queue definitions required. You can include this member in the CSQINP2 DD concatenation of the queue manager startup procedure, or you can use it as input to the COMMAND function in CSQUTIL utility to issue the required DEFINE commands.

There are two types of object definitions:

- SYSTEM.CHANNEL.xx, needed for any distributed queuing
- SYSTEM.CLUSTER.xx, needed for clustering

CSQ4INSJ system JMS object sample

Defines queues used in the JMS publish/subscribe domain.

CSQ4INSR object sample

Defines queues used by WebSphere Application Server and brokers.

CSQ4INYD object sample

If you are using distributed queuing and you need to set up your own queues, processes, and channels.

Sample data set thlqual.SCSQPROC(CSQ4INYD) contains sample definitions that you can use for customizing your distributed queuing objects. It comprises:

- A set of definitions for the sending end
- A set of definitions for the receiving end
- A set of definitions for using clients

You cannot use this sample as is -you must customize it before use. Then you can include this member in the CSQINP2 DD concatenation of the queue manager

startup procedure, or you can use it as input to the COMMAND function of the CSQUTIL utility to issue the required DEFINE commands. (This is preferable because it means that you don't have to redefine these objects each time you restart the queue manager).

CSQ4INYC object sample

If you are using clustering, definitions equivalent to the channel definitions and remote queue definitions of distributed queuing are created automatically, when needed. However, some manual channel definitions are needed – a cluster-receiver channel for the cluster and a cluster-sender definition to at least one cluster repository queue manager.

The sample data set: thlqual.SCSQPROC(CSQ4INYC) contains the following sample definitions that you can use for customizing your clustering objects:

- Definitions for the queue manager
- Definitions for the receiving channel
- Definitions for the sending channel
- Definitions for cluster queues
- Definitions for lists of clusters

You cannot use this sample as is - you must customize it before use. Then you can include this member in the CSQINP2 DD concatenation of the queue manager startup procedure, or you can use it as input to the COMMAND function of the CSQUTIL utility to issue the required DEFINE commands, this is preferable because it means that you don't have to redefine these objects each time that you restart WebSphere MQ.

CSQ4INYG object sample

The sample data set: thlqual.SCSQPROC(CSQ4INYG) contains the following sample definitions that you can use for customizing your own objects for general use:

- Dead-letter queue
- Default transmission queue
- CICS adapter objects

You cannot use this sample as is - you must customize it before use. Then you can include this member in the CSQINP2 DD concatenation of the queue manager startup procedure, or you can use it as input to the COMMAND function of the CSQUTIL utility to issue the required DEFINE commands. (This is preferable because it means that you don't have to redefine these objects each time that you restart WebSphere MQ).

In addition to the sample definitions here, you can use the system object definitions as the basis for your own resource definitions. For example, you can make a working copy of SYSTEM.DEFAULT.LOCAL.QUEUE and name it MY.DEFAULT.LOCAL.QUEUE. You can then change any of the parameters in this copy as required. You could then issue a DEFINE command by whichever method you choose, provided you have the authority to create resources of that type.

Default transmission queue:

Read "Default transmission queue" on page 38 before you decide whether you want to define a default transmission queue.

- If you decide that you do want to define a default transmission queue, remember that you must also define a channel to serve it.
- If you decide that you do not want to define one, remember to remove the DEFXMITQ statement from the ALTER QMGR command in the sample.

CICS adapter objects:

The sample defines an initiation queue named CICS01.INITQ. This queue is used by the WebSphere MQ-supplied CKTI transaction. You can change the name of this queue; however it must match the name specified in the CICS system initialization table (SIT) or SYSIN override in the INITPARM statement.

CSQ4INYS/CSQ4NQR object samples

Storage class definitions for using:

- one page set for each class of message
- multiple page sets for major classes of message

CSQ4DISP display sample

The sample data set: thlqual.SCSQPROC(CSQ4DISP) contains a set of generic DISPLAY commands that display all the defined resources on your queue manager. This includes the definitions for all WebSphere MQ objects and definitions such as storage classes and trace. These commands can generate a large amount of output. You can use this sample in the CSQINP2 data set or as input to the COMMAND function of the CSQUTIL utility.

CSQ4INPX sample

The sample data set: thlqual.SCSQPROC(CSQ4INPX) contains a set of commands that you might want to execute each time the channel initiator starts. You must customize this sample before use; you can then include it in the CSQINPX data set for the channel initiator.

CSQ4IVPQ and CSQ4IVPG samples

The sample data sets: thlqual.SCSQPROC(CSQ4IVPQ) and thlqual.SCSQPROC(CSQ4IVPG) contain sets of DEFINE commands that are required to run the installation verification programs (IVPs).

You can include these samples in the CSQINP2 data set. When you have run the IVPs successfully, you do not need to run them again each time the queue manager is restarted. Therefore, you do not need to keep these samples in the CSQINP2 concatenation permanently.

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 5. Where to find more information about system parameters and system objects

Topic	Where to look
Using initialization input data sets System parameter macros Installation verification program	WebSphere MQ for z/OS System Setup Guide
Utility programs	WebSphere MQ for z/OS System Administration Guide
MQSC commands	WebSphere MQ Script (MQSC) Command Reference
WebSphere MQ clusters	WebSphere MQ Queue Manager Clusters
WebSphere MQ events	Monitoring WebSphere MQ

Recovery and restart

WebSphere MQ for z/OS has robust features for restart and recovery which are described in this topic.

This chapter describes how a queue manager recovers after it has stopped, and what happens when it is restarted. It contains the following sections:

How changes are made to data

WebSphere MQ must interact with other subsystems to keep all the data consistent. This section discusses *units of recovery*; what they are and how they are used in *back outs*.

Units of recovery

A *unit of recovery* is the processing done by a single queue manager for an application program, that changes WebSphere MQ data from one point of consistency to another. A *point of consistency* – also called a *syncpoint* or *commit point* – is a point in time when all the recoverable data that an application program accesses is consistent.

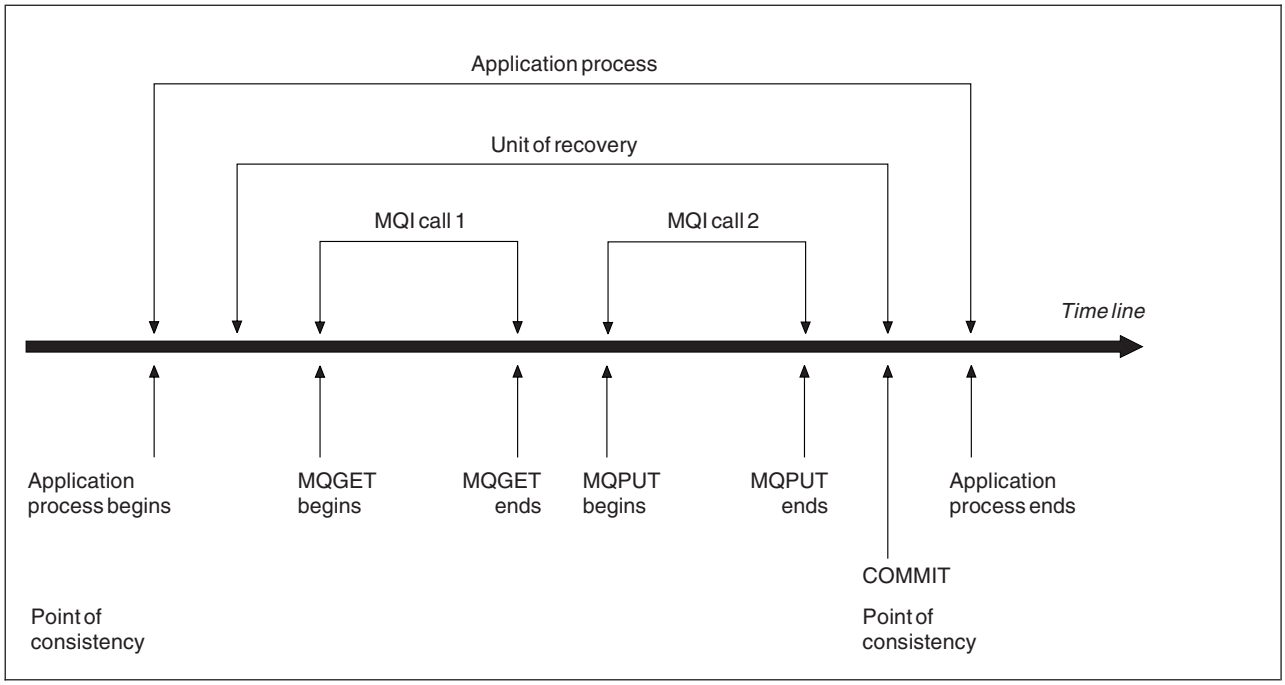


Figure 12. A unit of recovery within an application program. Typically, the unit of recovery consists of more than one MQI call. More than one unit of recovery can occur within an application program.

A unit of recovery begins with the first change to the data after the beginning of the program or following the previous point of consistency; it ends with a later point of consistency. Figure 12 shows the relationship between units of recovery, the point of consistency, and an application program. In this example, the application program makes changes to queues through MQI calls 1 and 2. The application program can include more than one unit of recovery or just one. However, any complete unit of recovery ends in a commit point.

For example, a bank transaction transfers funds from one account to another. First, the program subtracts the amount from the first account, account A. Then, it adds the amount to the second account, B. After subtracting the amount from A, the two accounts are inconsistent and WebSphere MQ cannot commit. They become consistent when the amount is added to account B. When both steps are complete, the program can announce a point of consistency through a commit, making the changes visible to other application programs.

Normal termination of an application program automatically causes a point of consistency. Some program requests in CICS and IMS programs also cause a point of consistency, for example, EXEC CICS SYNCPOINT.

Backing out work

If an error occurs within a unit of recovery, WebSphere MQ removes any changes to data, returning the data to its state at the start of the unit of recovery; that is, WebSphere MQ backs out the work. The events are shown in Figure 13 on page 46.

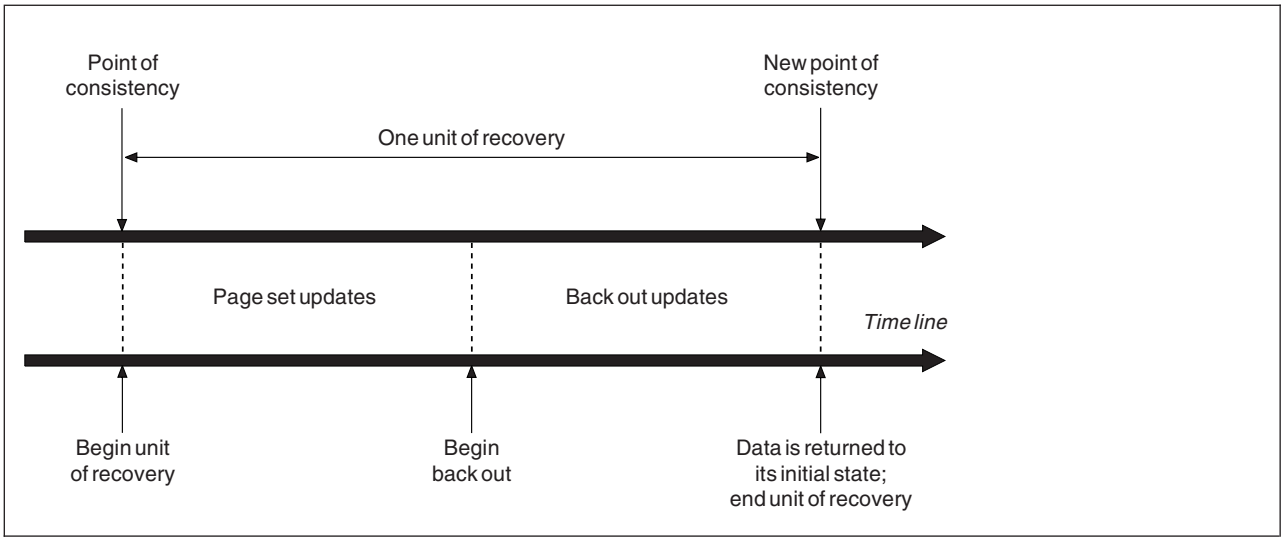


Figure 13. A unit of recovery showing back out

How consistency is maintained

If data in WebSphere MQ is to be consistent with batch, CICS, IMS, or TSO, any data changed in one must be matched by a change in the other. Before one system commits the changed data, it must know that the other system can make the corresponding change. So, the systems must communicate.

During a *two-phase commit* (for example under CICS), one subsystem coordinates the process. That subsystem is called the *coordinator*; the other is the *participant*. CICS or IMS is always the coordinator in interactions with WebSphere MQ, and WebSphere MQ is always the participant. In the batch or TSO environment, WebSphere MQ can participate in two-phase commit protocols coordinated by z/OS RRS.

During a *single-phase commit* (for example under TSO or batch), WebSphere MQ is always the coordinator in the interactions and completely controls the commit process.

In a WebSphere Application Server environment, the semantics of the JMS session object determine whether single-phase or two-phase commit coordination is used.

Consistency with CICS or IMS

The connection between WebSphere MQ and CICS or IMS supports the following syncpoint protocols:

- Two-phase commit – for transactions that update resources owned by more than one resource manager.

  This is the standard distributed syncpoint protocol. It involves more logging and message flows than a single-phase commit.
- Single-phase commit – for transactions that update resources owned by a single resource manager (WebSphere MQ).

  This protocol is optimized for logging and message flows.

- Bypass of syncpoint – for transactions that involve WebSphere MQ but which do nothing in the queue manager that requires a syncpoint (for example, browsing a queue).

In each case, CICS or IMS acts as the syncpoint manager.

The stages of the two-phase commit that WebSphere MQ uses to communicate with CICS or IMS are as follows:

1. In phase 1, each system determines independently whether it has recorded enough recovery information in its log, and can commit its work.

   At the end of the phase, the systems communicate. If they agree, each begins the next phase.
2. In phase 2, the changes are made permanent. If one of the systems abends during phase 2, the operation is completed by the recovery process during restart.

Illustration of the two-phase commit process:

Figure 14 illustrates the two-phase commit process. Events in the CICS or IMS coordinator are shown on the upper line, events in WebSphere MQ on the lower line.

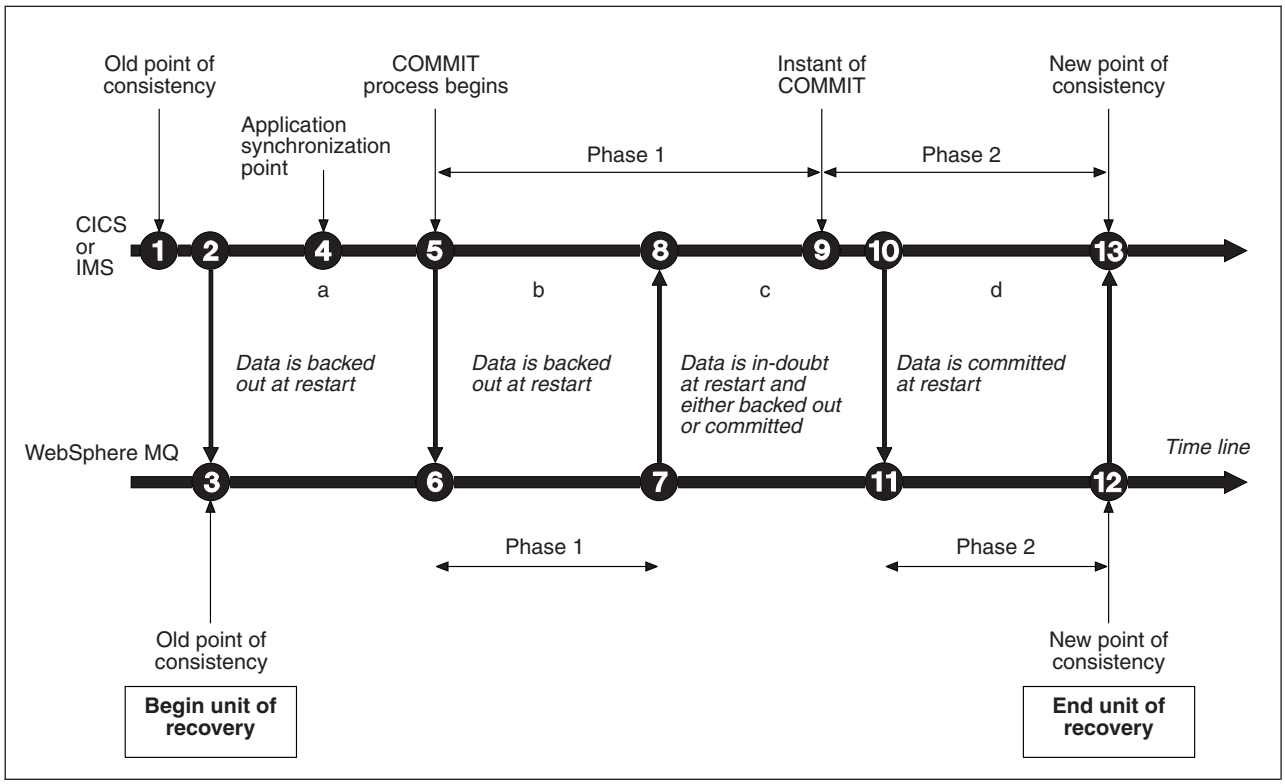


Figure 14. The two-phase commit process

The numbers in the following discussion are linked to those in the figure.

1. The data in the coordinator is at a point of consistency.
2. An application program in the coordinator calls WebSphere MQ to update a queue by adding a message.
3. This starts a unit of recovery in WebSphere MQ.

4. Processing continues in the coordinator until an application synchronization point is reached.
5. The coordinator then starts commit processing. CICS programs use a SYNCPOINT command or a normal application termination to start the commit. IMS programs can start the commit by using a CHKP call, a SYNC call, a GET UNIQUE call to the IOPCB, or a normal application termination. Phase 1 of commit processing begins.
6. As the coordinator begins phase 1 processing, so does WebSphere MQ.
7. WebSphere MQ successfully completes phase 1, writes this fact in its log, and notifies the coordinator.
8. The coordinator receives the notification.
9. The coordinator successfully completes its phase 1 processing. Now both subsystems agree to commit the data changes, because both have completed phase 1 and could recover from any errors. The coordinator records in its log the instant of commit – the irrevocable decision of the two subsystems to make the changes.

   The coordinator now begins phase 2 of the processing – the actual commitment.
10. The coordinator notifies WebSphere MQ to begin its phase 2.
11. WebSphere MQ logs the start of phase 2.
12. Phase 2 is successfully completed, and this is now a new point of consistency for WebSphere MQ. WebSphere MQ then notifies the coordinator that it has finished its phase 2 processing.
13. The coordinator finishes its phase 2 processing. The data controlled by both subsystems is now consistent and available to other applications.

How consistency is maintained after an abnormal termination

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

Figure 14 on page 47 shows four periods within the two phases: a, b, c, and d. The status of a unit of recovery depends on the period in which termination happened. The status can be one of the following:

In flight
: The queue manager ended before finishing phase 1 (period a or b); during restart, WebSphere MQ backs out the updates.

In doubt
: The queue manager ended after finishing phase 1 and before starting phase 2 (period c); only the coordinator knows whether the error happened before or after the commit (point 9). If it happened before, WebSphere MQ must back out its changes; if it happened after, WebSphere MQ must make its changes and commit them. At restart, WebSphere MQ waits for information from the coordinator before processing this unit of recovery.

In commit
: The queue manager ended after it began its own phase 2 processing (period d); it makes committed changes.

In backout
: The queue manager ended after a unit of recovery began to be backed out

but before the process was complete (not shown in the figure); during restart, WebSphere MQ continues to back out the changes.

What happens during termination

A queue manager terminates normally in response to the STOP QMGR command. If a queue manager stops for any other reason, the termination is abnormal.

Normal termination

In a normal termination, WebSphere MQ stops all activity in an orderly way. You can stop WebSphere MQ using either quiesce, force, or restart mode. The effects are given in Table 6.

Table 6. Termination using QUIESCE, FORCE, and RESTART

Thread type	QUIESCE	FORCE	RESTART
Active threads	Run to completion	Back out	Back out
New threads	Can start	Not permitted	Not permitted
New connections	Not permitted	Not permitted	Not permitted

Batch applications are notified if a termination occurs while the application is still connected.

With CICS, a current thread runs only to the end of the unit of recovery. With CICS, stopping a queue manager in quiesce mode stops the CICS adapter, and so if an active task contains more than one unit of recovery, the task does not necessarily run to completion.

If you stop a queue manager in force or restart mode, no new threads are allocated, and work on connected threads is rolled back. Using these modes can create in-doubt units of recovery for threads that are between commit processing phases. They are resolved when WebSphere MQ is reconnected with the controlling CICS, IMS, or RRS subsystem.

When you stop a queue manager, in any mode, the steps are:

1. Connections are ended.
2. WebSphere MQ ceases to accept commands.
3. WebSphere MQ ensures that any outstanding updates to the page sets are completed.
4. The DISPLAY USAGE command is issued internally by WebSphere MQ so that the restart RBA is recorded on the z/OS console log.
5. The shutdown checkpoint is taken and the BSDS is updated.

Terminations that specify quiesce mode do not affect in-doubt units of recovery. Any unit that is in doubt remains in doubt.

Abnormal termination

An abnormal termination can leave data in an inconsistent state, for example:

- A unit of recovery has been interrupted before reaching a point of consistency.
- Committed data has not been written to page sets.
- Uncommitted data has been written to page sets.

- An application program has been interrupted between phase 1 and phase 2 of the commit process, leaving the unit of recovery in doubt.

WebSphere MQ resolves any data inconsistencies arising from abnormal termination during restart and recovery.

What happens during restart and recovery

WebSphere MQ uses its recovery log and the bootstrap data set (BSDS) to determine what to recover when it restarts. The BSDS identifies the active and archive log data sets, and the location of the most recent WebSphere MQ checkpoint in the log.

After WebSphere MQ has been initialized, the queue manager restart process takes place as follows:

- Log initialization
- Current status rebuild
- Forward log recovery
- Backward log recovery
- Queue index rebuilding

When recovery has been completed:

- Committed changes are reflected in the data.
- In-doubt activity is reflected in the data. However, the data is locked and cannot be used until WebSphere MQ recognizes and acts on the in-doubt decision.
- Interrupted in-flight and in-abort changes have been removed from the queues. The messages are consistent and can be used.
- A new checkpoint has been taken.
- New indexes have been built for indexed queues containing persistent messages (described in "Rebuilding queue indexes" on page 52).

Batch applications are not notified when restart occurs *after* the application has requested a connection.

If dual BSDSs are in use, WebSphere MQ checks the consistency of the time stamps in the BSDS:

- If both copies of the BSDS are current, WebSphere MQ tests whether the two time stamps are equal. If they are not, WebSphere MQ issues message CSQJ120E and terminates. This can happen when the two copies of the BSDS are maintained on separate DASD volumes and one of the volumes was restored while the queue manager was stopped. WebSphere MQ detects the situation at restart.
- If one copy of the BSDS was de-allocated, and logging continued with a single BSDS, a problem could arise. If both copies of the BSDS are maintained on a single volume, and the volume was restored, or if both BSDS copies were restored separately, WebSphere MQ might not detect the restoration. In that case, log records not noted in the BSDS would be unknown to the system.

Understanding the log range required for recovery

During restart, the range of log data which must be read is dependent on many factors:

- At the time of an abnormal termination, there are typically many incomplete units of work in the system. As described earlier, restart processing will bring the system to a state of consistency, which may involve backing out inflight units of work, or recovering locks on indoubt units of work. Unit of work recovery requires that all unit of work log records for inflight, in-backout, and in-doubt units of work are available. WebSphere MQ will 'shunt' old units of work, so that unit of work recovery can be performed using a much smaller range of log data.
- At the time of an abnormal termination, there are typically many persistent updates which are only held in the buffer pool cache. They have not yet been written to disk. These changes must be read from the log, and reapplied to the data held in page sets. Page set recovery RBAs in the checkpoint describe the lowest log RBA required for updating the page sets to a consistent state.
- If old page sets have been introduced into the system, for example, a page set backup has been introduced to recover from a media failure, all the changes must be read from the log from the time the backup was taken. These changes are reapplied to the data held in the page set being recovered. Page set recovery RBAs held in page 0 of the page set describe the lowest log RBA required for media recovery of a page set.
- If using persistent messages on shared queues, a range of log data is required to recover CFSTRUCTs which are holding persistent messages. The earliest log data that would be required to perform a CFSTRUCT recovery, is from around the time of the old CFSTRUCT BACKUP.

During normal running, the DISPLAY USAGE TYPE(DATASET) command can be used to view the recovery log range associated with these factors (it is unable to provide information due to reintroducing old page sets, of course). A best practice would be to regularly monitor the values output from DISPLAY USAGE TYPE(DATASET) to identify any issues that might prolong a queue manager restart in the event of an abnormal termination.

In addition, the queue manager issues informational messages relating to these factors:

- CSQJ160I and CSQJ161I warn of long running units of work.
- CSQR026I and CSQR027I provide information about whether these long running units of work have been successfully shunted.
- CSQE040I and CSQE041E warn that structure backups are getting old, and consequently a RECOVER CFSTRUCT operation would take a long time.

Determining which application has a long running unit of work

It is possible to determine the application with the long-running unit of work. To do this, you use the DISPLAY CONN command.

The DISPLAY CONN command returns connection information for all the applications connected to the queue manager, together with additional information that helps you determine which application(s) currently have a long-running unit of work. The information returned by the DISPLAY CONN command is similar to the information returned by the DISPLAY QSTATUS command, but the main difference is that DISPLAY CONN displays information about objects, and transactional information for a particular connection, rather than details of which connections are associated with a particular object.

For each connected application, the DISPLAY CONN command returns the following information:

- Basic information including the Connection Id and PID.
- Transactional information for that connection, including the time and date when the transaction was created (that is, when the first MQGET/PUT was made under syncpoint), and when the transaction first wrote to the log.
- Log time information indicating which application still has a long running unit of work.
- A list of all objects that the connection currently has open. Details for each object are returned as a separate message, with the Connection Id used as a key. Because there are different types of object such as queues and queue managers, the information displayed with the object is specific to its object type.

Rebuilding queue indexes

To increase the speed of **MQGET** operations on a queue where messages are not retrieved sequentially, you can specify that you want WebSphere MQ to maintain an index of the message or correlation identifiers or groupid for all the messages on that queue (as described in the WebSphere MQ Application Programming Guide).

When a queue manager is restarted, these indexes are rebuilt for each queue. This applies only to persistent messages; nonpersistent messages are deleted at restart. If your indexed queues contain large numbers of persistent messages, this increases the time taken to restart the queue manager.

You can choose to have indexes rebuilt asynchronously to queue manager startup by using the QINDXBLD parameter of the CSQ6SYSP macro. If you set QINDXBLD=NOWAIT, WebSphere MQ restarts without waiting for indexes to rebuild.

How in-doubt units of recovery are resolved

If WebSphere MQ loses its connection to CICS, IMS, or RRS, it normally attempts to recover all inconsistent objects at restart. The information needed to resolve in-doubt units of recovery must come from the coordinating system. The next section describes the process of resolution.

How in-doubt units of recovery are resolved from CICS

The resolution of in-doubt units has no effect on CICS resources. CICS is in control of recovery coordination and, when it restarts, automatically commits or backs out each unit, depending on whether there was a log record marking the beginning of the commit. The existence of in-doubt objects does not lock CICS resources while WebSphere MQ is being reconnected.

One of the functions of the CICS adapter is to keep data synchronized between CICS and WebSphere MQ. If a queue manager abends while connected to CICS, it is possible for CICS to commit or back out work without WebSphere MQ being aware of it. When the queue manager restarts, that work is termed *in doubt*.

WebSphere MQ cannot resolve these in-doubt units of recovery (that is, commit or back out the changes made to WebSphere MQ resources) until the connection to CICS is restarted or reconnected.

A process to resolve in-doubt units of recovery is initiated during startup of the CICS adapter. The process starts when the adapter requests a list of in-doubt units of recovery. Then:

- The adapter receives a list of in-doubt units of recovery for this connection ID from WebSphere MQ, and passes them to CICS for resolution.
- CICS compares entries from this list with entries in its own log. CICS determines from its own list what action it took for each in-doubt unit of recovery.

Under some circumstances, CICS cannot run the WebSphere MQ process to resolve in-doubt units of recovery. When this happens, WebSphere MQ sends one of the following messages:

- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E

followed by the message CSQC408I.

For details of what these messages mean, see the WebSphere MQ for z/OS Messages and Codes manual.

For all resolved units, WebSphere MQ updates the queues as necessary and releases the corresponding locks. Unresolved units can remain after restart. Resolve them by the methods described in the WebSphere MQ for z/OS System Administration Guide.

How in-doubt units of recovery are resolved from IMS

Resolving in-doubt units of recovery in IMS has no effect on DL/I resources. IMS is in control of recovery coordination and, when it restarts, automatically commits or backs out incomplete DL/I work. The decision to commit or back out for online regions (non-fast-path) is on the presence or absence of IMS log record types X'3730' and X'3801' respectively. The existence of in-doubt units of recovery does not imply that DL/I records are locked until WebSphere MQ connects.

During queue manager restart, WebSphere MQ makes a list of in-doubt units of recovery. IMS builds its own list of residual recovery entries (RREs). The RREs are logged at IMS checkpoints until all entries are resolved.

During reconnection of an IMS region to WebSphere MQ, IMS indicates to WebSphere MQ whether to commit or back out units of work marked in WebSphere MQ as in doubt.

When in-doubt units are resolved:

1. If WebSphere MQ recognizes that it has marked an entry for commit and IMS has marked it to be backed out, WebSphere MQ issues message CSQQ010E. WebSphere MQ issues this message for all inconsistencies of this type between WebSphere MQ and IMS.
2. If WebSphere MQ has any remaining in-doubt units, the adapter issues message CSQQ008I.

For all resolved units, WebSphere MQ updates queues as necessary and releases the corresponding locks.

WebSphere MQ maintains locks on in-doubt work that was not resolved. This can cause a backlog in the system if important locks are being held. The connection remains active so you can resolve the IMS RREs. Recover the in-doubt threads by the methods described in the WebSphere MQ for z/OS System Administration Guide.

All in-doubt work should be resolved unless there are software or operating problems, such as with an IMS cold start. In-doubt resolution by the IMS control region takes place in two circumstances:

1. At the start of the connection to WebSphere MQ, during which resolution is done synchronously.
2. When a program abends, during which the resolution is done asynchronously.

How in-doubt units of recovery are resolved from RRS

One of the functions of the RRS adapter is to keep data synchronized between WebSphere MQ and other RRS-participating resource managers. If a failure occurs when WebSphere MQ has completed phase one of the commit and is waiting for a decision from RRS (the commit coordinator), the unit of recovery enters the in-doubt state.

When communication is reestablished between RRS and WebSphere MQ, RRS automatically commits or backs out each unit of recovery, depending on whether there was a log record marking the beginning of the commit. WebSphere MQ cannot resolve these in-doubt units of recovery (that is, commit or back out the changes made to WebSphere MQ resources) until the connection to RRS is reestablished.

Under some circumstances, RRS cannot resolve in-doubt units of recovery. When this happens, WebSphere MQ sends one of the following messages to the z/OS console:

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

For details of what these messages mean, see the WebSphere MQ for z/OS Messages and Codes manual.

For all resolved units of recovery, WebSphere MQ updates the queues as necessary and releases the corresponding locks. Unresolved units of recovery can remain after restart. Resolve them by the method described in the WebSphere MQ for z/OS System Administration Guide.

Shared queue recovery

This section describes WebSphere MQ recovery in the queue-sharing group environment.

Transactional recovery

When an application issues an **MQBACK** call or terminates abnormally (for example, because of an EXEC CICS ROLLBACK or an IMS abend) thread-level information stored in the queue manager ensures that the in-flight unit of work is

rolled back. **MQPUT** and **MQGET** operations within syncpoint on shared queues are rolled back in the same way as updates to non-shared queues.

Peer recovery

If a queue manager fails, it disconnects abnormally from the Coupling Facility structures that it is currently connected to. If the connection between the z/OS instance and the Coupling Facility fails (for example, physical link failure or power off of a Coupling Facility or partition) this is also detected as an abnormal termination of the connection between the queue manager and the Coupling Facility structures involved. Other queue managers in the same queue-sharing group that remain connected to that structure detect the abnormal disconnection and all attempt to initiate *peer recovery* for the failed queue manager on that structure. Only one of these queue managers initiates peer recovery successfully, but all the other queue managers cooperate in the recovery of units of work that were owned by the queue manager that failed.

If a queue manager fails when there are no peers connected to a structure, recovery is performed when another queue manager connects to that structure, or when the queue manager that failed restarts.

Peer recovery is performed on a structure by structure basis and it is possible for a single queue manager to participate in the recovery of more than one structure at the same time. However, the set of peers cooperating in the recovery of different structures might vary depending on which queue managers were connected to the different structures at the time of failure.

When the failed queue manager restarts, it reconnects to the structures that it was connected to at the time of failure, and recovers any remaining unresolved units of work that were not recovered by peer recovery.

Peer recovery is a multi-phase process. During the first phase, units of work that had progressed beyond the in-flight phase are recovered; this might involve committing messages for units of work that are in-commit and locking messages for units of work that are in-doubt. During the second phase, queues that had threads active against them in the failing queue manager are checked, uncommitted messages related to in-flight units of work are rolled back, and information about active handles on shared queues in the failed queue manager are reset. This means that WebSphere MQ resets any indicators that the failing queue manager had a shared queue open for input-exclusive, allowing other active queue managers to open the queue for input.

Shared queue definitions

The queue objects that represent the attributes of a shared queue are held in the shared DB2 repository used by the queue-sharing group. You should ensure that adequate procedures are in place for the backup and recovery of the DB2 tables used to hold WebSphere MQ objects. You can also use the WebSphere MQ CSQUTIL utility to create MQSC commands for replay into a queue manager to redefine WebSphere MQ objects, including shared queue and group definitions stored in DB2.

Logging

Queue sharing groups can support persistent messages, because the messages on shared queues can be logged in the queue manager logs.

Coupling Facility and structure failures

There are two types of failure that can be reported for a Coupling Facility (CF) structure: structure failure and loss of connectivity. Sysplex services for data sharing (XES) inform WebSphere MQ of a CF structure failure or a CF failure with a structure failure event. If XES creates a loss of connectivity event this does not necessarily indicate that there is a problem with the structure, it might be that there is no connection available to communicate with the structure. It is possible that not all of the queue managers receive a loss of connectivity event for the structure; it depends on the configuration of connections to the CF. A loss of connectivity event can also be received because of operator commands, for example VARY PATH OFFLINE or CONFIG CHP OFFLINE.

The CF structures that are used by WebSphere MQ can be configured to use system-managed duplexing. This means that if there is a single failure, system-managed failover processing hides the failure of a structure or the loss of connectivity, and the queue manager is not informed of the failure. If there is a failure of both instances of a duplexed structure or connection, the queue manager receives the appropriate event and handles it in the same way as a failure event for a simplex structure. Details of how the queue manager handles the events are described in “Scenarios ” on page 57.

In the unlikely event of a CF or structure failure, any nonpersistent messages stored in the affected application structures are lost. You can recover persistent messages using the RECOVER CFSTRUCT command. If a recoverable application structure has failed, any further application activity to this structure is prevented until the structure has been recovered.

To ensure that you can recover a CF structure in a reasonable period of time, take frequent backups, using the BACKUP CFSTRUCT command. You can choose to perform the backups on any queue managers in the queue-sharing group or dedicate one queue manager to perform all the backups. Automate the process of taking backups to ensure that they are taken on a regular basis.

Each backup is written to the active log data set of the queue manager taking the backup. The shared queue DB2 repository records the name of the CF structure being backed up, the name of the queue manager doing the backup, the RBA range for this backup on that queue manager’s log, and the backup time.

The administration structure contains information about incomplete units of work on shared queues at the time of any application structure failure so the administration structure must be available during RECOVER CFSTRUCT processing. If the administration structure has failed, all the queue managers in the queue-sharing group must have rebuilt their administration structure entries before you can issue the RECOVER CFSTRUCT command. Version 6 and later queue managers might be able to do this without terminating (depending on the type of failure), otherwise the queue manager rebuilds its administration structure entries when it is started.

To recover an application structure, issue a RECOVER CFSTRUCT command to the queue manager that you want to perform the recovery. You can recover a single CF structure or you can recover several CF structures simultaneously. You can recover any queue manager in the queue-sharing group, it does not have to be the one that performed the backup, or one that has been connected previously to the failed structure. The RECOVER CFSTRUCT command uses the backup, located through the DB2 repository information (DB2 must therefore be available on the queue

manager where recovery is being carried out), and recovers this to the point of failure. The RECOVER CFSTRUCT command does this by applying log records from every queue manager in the queue-sharing group that has performed an MQPUT or MQGET between the start of the backup and the time of failure, to any shared queue that maps to the CF structure. The resulting merging of the logs might require reading a considerable amount of log data because all the log data written by participating queue managers since the backup will be read. You are strongly recommended to make frequent (for example, hourly) backups, especially if there are large messages within the backup.

Scenarios

If a failure is reported for a CF structure, the action taken by connected queue managers depends on the following:

- The type of failure reported by the XES component of z/OS to WebSphere MQ.
- The structure type (application or administration)
- The queue manager level (Version 5.3.1, Version 6.0, or Version 7.0)
- The CFLEVEL of the MQ CFSTRUCT object (2, 3, or 4. This is not the CFLEVEL of the CFCC microcode)

The following scenarios describe what happens when a failure is reported for the administration structure:

- If a structure failure event is received for the administration structure and the queue manager is running at Version 6.0 and later, the structure is reallocated and rebuilt automatically without the queue manager terminating. Because a structure failure has occurred the structure is not allocated in the CF, it is allocated by XES when a queue manager attempts to connect to it. When the queue manager has connected to the new instance of the structure, the queue manager writes only the entries for itself into the structure. This processing is carried out by the queue manager and is not part of XES rebuild processing.

  Any serialized applications that have already connected to the queue manager can continue processing. Any serialized application attempting to connect with the MQCNO_SERIALIZE_CONN_TAG_QSG or MQCNO_RESTRICT_CONN_TAG_QSG parameters receive the MQRC_CONN_TAG_NOT_USABLE return code until all the queue managers in the queue-sharing group have rebuilt their administration structure entries. Certain actions on the shared queue are suspended until the queue manager has reconnected to the administration structure and finished rebuilding the entries in the structure. The suspended actions include the following:

  – Opening and closing of shared queues.
  – Committing or backing out units of recovery.
  – Serialized applications connecting to or disconnecting from the queue manager.

  When the administration structure entries for the queue manager have been rebuilt, the suspended actions are resumed.

  You cannot back up or recover an application structure until all the queue managers in the queue-sharing group have rebuilt their administration structure entries. If a queue manager was not running at the time of the failure, or terminates before recovery of its part of the administration structure has been completed, restart this queue manager so that it can complete the rebuild of its part of the structure.
- If a loss of connectivity event is received for the administration structure, the queue manager terminates, irrespective of the version at which the queue

manager is running. When connectivity to the CF is possible again, restart the queue manager so that it can rebuild its part of the administration structure.

The following scenarios describe what happens when a failure is reported for an application structure:

- If a structure failure event is received for the administration structure and the queue manager is running at Version 5.3.1 or earlier, the queue manager terminates. Restart the queue manager so that it can rebuild its part of the administration structure.
- If a structure failure event is received for an application structure, and the CFLEVEL is 1 or 2, the queue manager terminates. Restart the queue manager. The first queue manager to attempt to connect to the structure again will cause XES to allocate a new instance of the structure.
- If a structure failure event is received for an application structure, and the CFLEVEL is 3 or 4, the queue managers connected to the structure continue to run. Applications that do not use the queues in the failed structure can continue normal processing. However, applications that attempt operations on queues in the failed structure receive an MQRC_CF_STRUC_FAILED error until the RECOVER CFSTRUCT command has successfully rebuilt the failed structure, at which point the application can open the queues again.
- If a loss of connectivity event is received for an application structure for any CFLEVEL, the queue manager terminates to allow other queue managers in the queue-sharing group that might still have connectivity to perform peer level recovery, and so make messages more available. Restart the queue manager that has failed and the connection to the structure will be reestablished when possible.

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 7. Where to find more information about recovery and restart

Topic	Where to look
Planning your backup strategy	"Planning for backup and recovery" on page 126
System parameters	WebSphere MQ for z/OS System Setup Guide
Routine backup and recovery procedures Resolving in-doubt threads Resolving problems during restart Utility programs	WebSphere MQ for z/OS System Administration Guide
Console messages	WebSphere MQ for z/OS Messages and Codes
MQSC commands	WebSphere MQ Script (MQSC) Command Reference

Security

This chapter discusses WebSphere MQ security. It contains the following sections:

Why you need to protect WebSphere MQ resources

Because WebSphere MQ handles the transfer of information that is potentially valuable, it needs the safeguard of a security system. This is to ensure that the resources WebSphere MQ owns and manages are protected from unauthorized access that might lead to the loss or disclosure of the information. It is essential that none of the following are accessed or changed by any unauthorized user or process:

- Connections to WebSphere MQ
- WebSphere MQ objects such as queues, processes, and namelists
- WebSphere MQ transmission links
- WebSphere MQ system control commands
- WebSphere MQ messages
- Context information associated with messages

To provide the necessary security, WebSphere MQ uses the z/OS system authorization facility (SAF) to route authorization requests to an External Security Manager (ESM), for example Security Server (previously known as RACF). WebSphere MQ does no security verification of its own. Where distributed queuing or clients are being used, you might require additional security measures, for which WebSphere MQ provides channel exits, the MCAUSER channel attribute, and the Secure Sockets Layer (SSL).

The decision to allow access to an object is made by the ESM and WebSphere MQ follows that decision. If the ESM cannot make a decision, WebSphere MQ prevents access to the object.

If you do nothing

If you do nothing about security, the most likely effect is that ***all*** users can access and change ***every*** resource. This includes not only local users, but also those on remote systems using distributed queuing or clients, where the logon security controls might be less strict than is normally the case for z/OS.

To enable security checking you must do the following:

- Install and activate an ESM (for example, Security Server).
- Define the MQADMIN class if you are using an ESM other than Security Server.
- Activate the MQADMIN class.

You need to consider whether or not using mixed-case resource names would be beneficial to your enterprise. If you do use mixed-case resource names in your ESM profiles you need to define and activate the MXADMIN class.

Security controls and options

You can specify whether security is turned on for the whole WebSphere MQ subsystem, and whether you want to perform security checks at queue manager or queue-sharing group level. You can also control the number of user IDs checked for API-resource security.

Subsystem security

Subsystem security is a control that specifies whether any security checking is done for the whole queue manager. If you do not require security checking (for example, on a test system), or if you are satisfied with the level of security on all the resources that can connect to WebSphere MQ (including clients and channels), you can turn security checking off for the queue manager or queue-sharing group so that no further security checking takes place.

This is the only check that can turn security off completely and determine whether any other security checks are performed or not. That is, if you turn off checking for the queue manager or queue-sharing group, no other WebSphere MQ checking is done; if you leave it turned on, WebSphere MQ checks your security requirements for other WebSphere MQ resources.

You can also turn security on or off for particular sets of resources, such as commands.

Queue manager or queue-sharing group level checking

You can implement security at queue manager level or at queue-sharing group level. If you implement security at queue-sharing group level, all the queue managers in the group share the same profiles. This means that there are fewer profiles to define and maintain, making security management easier. It also makes it easy to add a new queue manager to the queue-sharing group because it inherits the existing security profiles.

It is also possible to implement a combination of both if your installation requires it, for example, during migration or if you have one queue manager in the queue-sharing group that requires different levels of security to the other queue managers in the group.

Queue-sharing group level security

Queue-sharing group level security checking is performed for the entire queue-sharing group. It enables you to simplify security administration because it requires you to define fewer security profiles. The authorization of a user ID to use a particular resource is handled at the queue-sharing group level, and is independent of which queue manager that user ID is using to access the resource.

For example, say a server application runs under user ID SERVER and wants access to a queue called SERVER.REQUEST, and you want to run an instance of SERVER on each z/OS image in the sysplex. Rather than permitting SERVER to open SERVER.REQUEST on each queue manager individually (queue manager level security), you can permit access only at the queue-sharing group level.

You can use queue-sharing group level security profiles to protect all types of resource, whether local or shared.

Queue manager level security
You can use queue manager level security profiles to protect all types of resource, whether local or shared.

Combination of both levels
You can use a combination of both queue manager and queue-sharing group level security.

You can override queue-sharing group level security settings for a particular queue manager that is a member of that group. This means that you can perform a different level of security checks on an individual queue manager to those performed on the other queue managers in the group.

Controlling the number of user IDs checked

RESLEVEL is a Security Server profile that controls the number of user IDs checked for WebSphere MQ resource security. Normally, when a user attempts to access a WebSphere MQ resource, Security Server checks the relevant user ID or IDs to see if access is allowed to that resource. By defining a RESLEVEL profile you can control whether zero, one or, where applicable, two user IDs are checked.

These controls are done on a connection by connection basis, and last for the life of the connection.

There is only one RESLEVEL profile for each queue manager. Control is implemented by the access that a user ID has to this profile.

Mixed case or uppercase WebSphere MQ RACF classes

You can now use mixed case RACF profile support, which allows you to use mixed case resource names and define WebSphere MQ RACF profiles to protect them.

You can choose to either:

- Continue using uppercase only WebSphere MQ RACF Classes as in previous releases, or
- Use the new mixed case WebSphere MQ RACF classes.

Without the use of mixed case RACF profiles, you can still use mixed case resource names in WebSphere MQ for z/OS; however, these resource names can only be protected by generic RACF profiles in the uppercase WebSphere MQ classes. When using mixed case WebSphere MQ RACF profile support you can provide a more granular level of protection by defining WebSphere MQ RACF profiles in the mixed case WebSphere MQ classes.

Resources you can protect

When a queue manager starts, or when instructed by an operator command, WebSphere MQ determines which resources you want to protect. You can control which security checks are performed for each individual queue manager. For example, you could implement a number of security checks on a production queue manager, but none on a test queue manager.

Connection security

Connection security checking is carried out either when an application program tries to connect to a queue manager by issuing an **MQCONN** or **MQCONNX** request, or when the channel initiator, or CICS or IMS adapter issues a connection request.

If you are using queue manager level security, you can turn connection security checking off for a particular queue manager, but if you do any user can connect to that queue manager.

For the CICS adapter, only the CICS address space user ID is used for the connection security check, not the individual CICS terminal user ID. For the IMS adapter, when the IMS control or dependent regions connect to WebSphere MQ, the IMS address space user ID is checked. For the channel initiator, the user ID used by the channel initiator address space is checked.

You can turn connection security checking on or off at either queue manager or queue-sharing group level.

API-resource security

Resources are checked when an application opens an object with an **MQOPEN** or an **MQPUT1** call. The access needed to open an object depends on what open options are specified when the queue is opened.

API-resource security is subdivided into the following checks:

- Queue
- Process
- Namelist
- Alternate user
- Context

No security checks are performed when opening the queue manager object or when accessing storage class objects.

Queue security:

Queue security checking controls who is allowed to open which queue, and what options they are allowed to open it with. For example, a user might be allowed to open a queue called PAYROLL.INCREASE.SALARY to browse the messages on the queue (using the MQOO_BROWSE option), but not to remove messages from the queue (using one of the MQOO_INPUT_* options). If you turn checking for queues off, any user can open any queue with any valid open option (that is, any valid MQOO_* option on an **MQOPEN** or **MQPUT1** call).

You can turn queue security checking on or off at either queue manager or queue-sharing group level.

Process security:

Process security checking is carried out when a user opens a process definition object. If you turn checking for processes off, any user can open any process.

You can turn process security checking on or off at either queue manager or queue-sharing group level.

Namelist security:

Namelist security checking is carried out when a user opens a namelist. If you turn checking for namelists off, any user can open any namelist.

You can turn namelist security checking on or off at either queue manager or queue-sharing group level.

Alternate user security:

Alternate user security controls whether one user ID can use the authority of another user ID to open a WebSphere MQ object.

For example:

- A server program running under user ID PAYSERV retrieves a request message from a queue that was put on the queue by user ID USER1.
- When the server program gets the request message, it processes the request and puts the reply back into the reply-to queue specified with the request message.
- Instead of using its own user ID (PAYSERV) to authorize opening the reply-to queue, the server can specify some other user ID, in this case, USER1. In this example, alternate user security would control whether user ID PAYSERV is allowed to specify user ID USER1 as an alternate user ID when opening the reply-to queue.

The alternate user ID is specified in the *AlternateUserId* field of the object descriptor (MQOD).

You can use alternate user IDs on any WebSphere MQ object, for example, processes or namelists. It does not affect the user ID used by any other resource managers, for example, for CICS security or for z/OS data set security.

If alternate user security is not active, any user can use any other user ID as an alternate user ID.

You can turn alternate user security checking on or off at either queue manager or queue-sharing group level.

Context security:

Context is information that is applicable to a particular message and is contained in the message descriptor (MQMD) that is part of the message. The context information comes in two sections:

Identity section
The user of the application that first put the message to a queue. It consists of the following fields:

- *UserIdentifier*
- *AccountingToken*
- *ApplIdentityData*

Origin section
The application that put the message on the queue where it is currently stored. It consists of the following fields:

- *PutApplType*
- *PutApplName*
- *PutDate*
- *PutTime*
- *ApplOriginData*

Applications can specify the context data when either an **MQPUT** or an **MQPUT1** call is made. The application might generate the data, the data might be passed on from another message, or the queue manager might generate the data by default. For example, server programs can use context data to check the identity of the requester, that is, did this message come from the correct application? Typically, the *UserIdentifier* field is used to determine the user ID of an alternate user.

You use context security to control whether the user can specify any of the context options on any **MQOPEN** or **MQPUT1** call. For information about the context options, see the WebSphere MQ Application Programming Guide; for descriptions of the message descriptor fields relating to context, see the WebSphere MQ Application Programming Reference manual.

If you turn context security checking off, any user can use any of the context options that the queue security allows.

You can turn context security checking on or off at either queue, queue manager or queue-sharing group level.

Command security

Command security checking is carried out when a user issues an MQSC command from any of the sources described in "Issuing commands" on page 69. You can make a separate check on the resource specified by the command as described in "Command resource security."

If you turn off command checking, issuers of commands are not checked to see whether they have the authority to issue the command.

If MQSC commands are entered from a console, the console must have the z/OS SYS console authority attribute. Commands that are issued from the CSQINP1 or CSQINP2 data sets, or internally by the queue manager, are exempt from all security checking while those for CSQINPX use the user ID of the channel initiator address space. You should control who is allowed to update these data sets through normal data set protection.

You can turn command security checking on or off at either queue manager or queue-sharing group level.

Command resource security

Some MQSC commands, for example defining a local queue, involve the manipulation of WebSphere MQ resources. When command resource security is active, each time a command involving a resource is issued, WebSphere MQ checks to see if the user is allowed to change the definition of that resource.

You can use command resource security to help enforce naming standards. For example, a payroll administrator might be allowed to delete and define only queues with names beginning "PAYROLL". If command resource security is

inactive, no security checks are made on the resource that is being manipulated by the command. Do not confuse command resource security with command security; the two are independent.

Turning off command resource security checking does not affect the resource checking that is done specifically for other types of processing that do not involve commands.

You can turn command resource security checking on or off at either queue manager or queue-sharing group level.

Channel security

When you are using channels, the security features available depend on which communications protocol you are going to use. If you use TCP, there are no security features provided with the communications protocol, although you can use SSL. If you are using APPC, you can flow user ID information from the sending MCA through the network to the destination MCA for verification.

For both protocols, you can specify which user IDs you want to check for security purposes, and how many. Again, the choices available to you depend on which protocol you are using, what you specify when you define the channel, and the RESLEVEL settings for the channel initiator.

You can also write your own security exit programs to be called by the MCA.

Secure Sockets Layer (SSL)

You can also use the Secure Sockets Layer (SSL) to provide channel security. SSL support is provided with WebSphere MQ for z/OS. SSL is an industry-standard protocol that provides a data security layer between application protocols and the communications layer, usually TCP/IP. SSL uses encryption techniques, digital signatures and digital certificates to provide message privacy, message integrity and mutual authentication between clients and servers.

For a detailed explanation of SSL, see the WebSphere MQ Security book.

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 8. Where to find more information about security

Topic	Where to look
Setting up your security Channel security	WebSphere MQ for z/OS System Setup Guide
Console messages	WebSphere MQ for z/OS Messages and Codes
MQSC commands Defining channels	WebSphere MQ Script (MQSC) Command Reference
Channel security exits	WebSphere MQ Intercommunication

Table 8. Where to find more information about security (continued)

Topic	Where to look
Secure Sockets Layer (SSL)	WebSphere MQ Security

Availability

WebSphere MQ for z/OS has many features for high availability. This topic describes some of the considerations for availability.

There are several features of WebSphere MQ that are designed to increase system availability if the queue manager or channel initiator fails. These are discussed in the following sections:

Sysplex considerations

Improve Websphere MQ availability in a sysplex.

In a *sysplex* a number of z/OS operating system images collaborate in a single system image and communicate using a Coupling Facility. WebSphere MQ can use the facilities of the sysplex environment for enhanced availability.

Removing the affinities between a queue manager and a particular z/OS image allows a queue manager to be restarted on a different z/OS image in the event of an image failure. The restart mechanism can be manual, use ARM, or use system automation, if you ensure the following:

- All page sets, logs, bootstrap data sets, code libraries, and queue manager configuration data sets are defined on shared volumes.
- The subsystem definition has sysplex scope and a unique name within the sysplex.
- The level of *early code* installed on every z/OS image at IPL time is at the same level.
- TCP virtual IP addresses (VIPA) is available on each TCP stack in the sysplex, and you have configured WebSphere MQ TCP listeners and inbound connections to use VIPAs rather than default host names.

For more information about using TCP in a sysplex, see *TCP/IP in a sysplex*, SG24–5235, an IBM® Redbooks® publication.

You can additionally configure multiple queue managers running on different operating system images in a sysplex to operate as a queue-sharing group, which can take advantage of shared queues and shared channels for higher availability and workload balancing.

Shared queues

In the queue-sharing group environment, an application can connect to any of the queue managers within the queue-sharing group. Because all the queue managers in the queue-sharing group can access the same set of shared queues, the application does not depend on the availability of a particular queue manager; any queue manager in the queue-sharing group can service any queue. This gives greater availability if a queue manager stops because all the other queue managers

in the queue-sharing group can continue processing the queue. For information about high availability of shared queues, see “Advantages of using shared queues” on page 13.

To further enhance the availability of messages in a queue-sharing group, WebSphere MQ detects if another queue manager in the group disconnects from the Coupling Facility abnormally, and completes units of work for that queue manager that are still pending, where possible. This is known as *peer recovery* and is described in “Peer recovery” on page 55.

Peer recovery cannot recover units of work that were in doubt at the time of the failure. You can use the Automatic Restart Manager (ARM) to restart all the systems involved in the failure (CICS, DB2, and WebSphere MQ for example), and to ensure that they are all restarted on the same new processor. This means that they can resynchronize, and gives rapid recovery of in-doubt units of work. This is described in “Using the z/OS Automatic Restart Manager (ARM)” on page 68.

Shared channels

In the queue-sharing group environment, WebSphere MQ provides functions that give high availability to the network. The channel initiator enables you to use networking products that balance network requests across a set of eligible servers and hide server failures from the network (for example, VTAM® generic resources). WebSphere MQ uses a generic port for inbound requests so that attach requests can be routed to any available channel initiator in the queue-sharing group. This is described in “Shared channels” on page 16.

Shared outbound channels take the messages they send from a shared transmission queue. Information about the status of a shared channel is held in one place for the whole queue-sharing group level. This means that a channel can be restarted automatically on a different channel initiator in the queue-sharing group if the channel initiator, queue manager, or communications subsystem fails. This is called *peer channel recovery* and is described in “Shared outbound channels” on page 17.

WebSphere MQ network availability

WebSphere MQ messages are carried from queue manager to queue manager in a WebSphere MQ network using channels. You can change the configuration at a number of levels to improve the network availability of a queue manager, and the ability of a WebSphere MQ channel to detect a network problem and to reconnect.

TCP *Keepalive* is available for TCP/IP channels. It causes TCP to send packets periodically between sessions to detect network failures. The KAINT channel attribute determines the frequency of these packets for a channel.

AdoptMCA allows a channel, blocked in receive processing as a result of a network outage, to be terminated and replaced by a new connection request. You control AdoptMCA using the ADOPTMCA channel initiator parameter, set by the CSQ6CHIP macro.

ReceiveTimeout prevents a channel from being permanently blocked in a network receive call. The RCVTIME and RCVTMIN channel initiator parameters, set by the CSQ6CHIP macro, determine the receive timeout characteristics for channels, as a function of their heartbeat interval.

Using the z/OS Automatic Restart Manager (ARM)

You can use WebSphere MQ for z/OS in conjunction with the z/OS automatic restart manager (ARM). If a queue manager or a channel initiator has failed, ARM restarts it on the same z/OS image. If z/OS fails, a whole group of related subsystems and applications also fail. ARM can restart all the failed systems automatically, in a predefined order, on another z/OS image within the sysplex. This is called a cross-system restart.

ARM enables rapid recovery of in-doubt transactions in the shared queue environment. It also gives higher availability if you are not using queue-sharing groups.

You can use ARM to restart a queue manager on a different z/OS image within the sysplex in the event of z/OS failure.

To enable automatic restart, you must do the following:

1. Set up an ARM coupling data set.
2. Define the automatic restart actions that you want z/OS to perform in an *ARM policy*.
3. Start the ARM policy.

If you want to restart queue managers in different z/OS images automatically, every queue manager in each z/OS image on which that queue manager might be restarted must be defined with a sysplex-wide unique 4-character subsystem name.

Using ARM with WebSphere MQ is described in the WebSphere MQ for z/OS System Administration Guide.

Using the z/OS Extended Recovery Facility (XRF)

You can use WebSphere MQ in an extended recovery facility (XRF) environment. All WebSphere MQ-owned data sets (executable code, BSDSs, logs, and page sets) must be on DASD shared between the active and alternate XRF processors.

If you use XRF for recovery, you must stop the queue manager on the active processor and start it on the alternate. For CICS, you can do this using the command list table (CLT) provided by CICS, or the system operator can do it manually. For IMS, this is a manual operation and you must do it after the coordinating IMS system has completed the processor switch.

WebSphere MQ utilities must be completed or terminated before the queue manager can be switched to the alternate processor. Consider the effect of this potential interruption carefully when planning your XRF recovery plans.

Take care to prevent the queue manager starting on the alternate processor before the queue manager on the active processor terminates. A premature start can cause severe integrity problems in data, the catalog, and the log. Using global resource serialization (GRS) helps avoid the integrity problems by preventing simultaneous use of WebSphere MQ on the two systems. You must include the BSDS as a protected resource, and you must include the active and alternate XRF processors in the GRS ring.

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 9. Where to find more information about availability

Topic	Where to look
Queue-sharing groups	"Shared queues and queue-sharing groups" on page 9
System parameters	WebSphere MQ for z/OS System Setup Guide
Using the Automatic Restart Manager Utility programs	WebSphere MQ for z/OS System Administration Guide
Console messages	WebSphere MQ for z/OS Messages and Codes
MQSC commands	WebSphere MQ Script (MQSC) Command Reference

Commands

This chapter discusses how to use the MQSC commands, PCF commands, and WebSphere MQ for z/OS utilities to create objects and manage your queue managers. It includes the following sections:

- "Issuing commands"
- "The WebSphere MQ for z/OS utilities" on page 77
- "Where to find more information" on page 79

Issuing commands

WebSphere MQ for z/OS supports MQSC commands, which can be issued from the following sources:

- The z/OS console or equivalent (such as SDSF/TSO).
- The initialization input data sets.
- The supplied batch utility, CSQUTIL, processing a list of commands in a sequential data set.
- A suitably authorized application, by sending a command as a message to the command input queue. The application can be any of the following:
  - A batch region program
  - A CICS application
  - An IMS application
  - A TSO application
  - An application program or utility on another WebSphere MQ system

Table 11 on page 73 summarizes the MQSC commands and the sources from which they can be issued.

Much of the functionality of these commands is available in a user-friendly way from the WebSphere MQ for z/OS operations and controls panels.

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

WebSphere MQ for z/OS also supports Programmable Command Format (PCF) commands. These simplify the creation of applications for the administration of WebSphere MQ. MQSC commands are in human-readable text form, whereas PCF enables applications to create requests and read the replies without having to parse text strings. Like MQSC commands, applications issue PCF commands by sending them as messages to the command input queue. For more information about using PCF commands and for details of the commands, see the *WebSphere MQ Programmable Command Formats and Administration Interface* book.

Private and global definitions

When you define an object on WebSphere MQ for z/OS, you can choose whether you want to share that definition with other queue managers (a *global* definition), or whether the object definition is to be used by one queue manager only (a *private* definition). This is called the object *disposition*.

Global definition

If your queue manager belongs to a queue-sharing group, you can choose to share any object definitions you make with the other members of the group. This means that you have to define an object once only, reducing the total number of definitions required for the whole system.

Global object definitions are held in a *shared repository* (a DB2 shared database), and are available to all the queue managers in the queue-sharing group. These objects have a disposition of GROUP.

Private definition

If you want to create an object definition that is required by one queue manager only, or if your queue manager is not a member of a queue-sharing group, you can create object definitions that are not shared with other members of a queue-sharing group.

Private object definitions are held on page set zero of the defining queue manager. These objects have a disposition of QMGR.

You can create private definitions for all types of WebSphere MQ objects except CF structures (that is, channels, namelists, process definitions, queues, queue managers, storage class definitions, and authentication information objects), and global definitions for all types of objects except queue managers.

WebSphere MQ automatically copies the definition of a group object to page set zero of each queue manager that uses it. You can alter the copy of the definition temporarily if you want, and WebSphere MQ allows you to refresh the page set copies from the repository copy if required. WebSphere MQ always tries to refresh the page set copies from the repository copy on start up (for channel commands, this is done when the channel initiator restarts), or if the group object is changed. This ensures that the page set copies reflect the version on the repository, including any changes that were made when the queue manager was inactive. The copies are refreshed by generating DEFINE REPLACE commands, therefore there are circumstances under which the refresh is not performed, for example:

- If a copy of the queue is open, a refresh that changes the usage of the queue fails.
- If a copy of a queue has messages on it, a refresh that deletes that queue fails.

- If a copy of a queue would require ALTER with FORCE to change it.

In these circumstances, the refresh is not performed on that copy, but is performed on the copies on all other queue managers.

If the queue manager is shut down and then restarted stand-alone, any local copies of objects are deleted, unless for example, the queue has associated messages.

There is a third object disposition that applies to local queues only. This allows you to create shared queues. The definition for a shared queue is held on the shared repository and is available to all the queue managers in the queue-sharing group. In addition, the messages on a shared queue are also available to all the queue managers in the queue sharing group. This is described in "Shared queues and queue-sharing groups" on page 9. Shared queues have an object disposition of SHARED.

The following table summarizes the effect of the object disposition options for queue managers started stand-alone, and as a member of a queue-sharing group.

Disposition	Stand-alone queue manager	Member of a queue-sharing group
QMGR	Object definition held on page set zero.	Object definition held on page set zero.
GROUP	Not allowed.	Object definition held in the shared repository. Local copy held on page set zero of each queue manager in the group.
SHARED	Not allowed.	Queue definition held in the shared repository. Messages available to any queue manager in the group.

Manipulating global definitions:

If you want to change the definition of an object that is held in the shared repository, you need to specify whether you want to change the version on the repository, or the local copy on page set zero. Use the object disposition as part of the command to do this.

Directing commands to different queue managers

You can choose to execute a command on the queue manager where it is entered, or on a different queue manager in the queue-sharing group. You can also choose to issue a particular command in parallel on all the queue managers in a queue-sharing group. This is possible for both MQSC commands and PCF commands.

This is determined by the *command scope*. The command scope is used in conjunction with the object disposition to determine which version of an object you want to work with.

For example, you might want to alter some of the attributes of an object, the definition of which is held in the shared repository.

- You might want to change the version on one queue manager only, and not make any changes to the version on the repository or those in use by other queue managers.

- You might want to change the version in the shared repository for future users, but leave existing copies unchanged.
- You might want to change the version in the shared repository, but also want your changes to be reflected immediately on all the queue managers in the queue-sharing group that hold a copy of the object on their page set zero.

Use the command scope to specify whether the command is executed on this queue manager, another queue manager, or all queue managers. Use the object disposition to specify whether the object you are manipulating is in the shared repository (a group object), or is a local copy on page set zero (a queue manager object).

You do not have to specify the command scope and object disposition to work with a shared queue because every queue manager in the queue-sharing group sees the shared queue as a single queue.

Command summary

Table 10 summarizes the MQSC and PCF commands that are available on WebSphere MQ for z/OS to alter, define, delete and display WebSphere MQ objects.

Table 10. Summary of the main MQSC and PCF commands by object type

MQSC command	**ALTER**	**DEFINE**	**DISPLAY**	**DELETE**
PCF command	**Change**	**Create/Copy**	**Inquire**	**Delete**
AUTHINFO	X	X	X	X
CFSTATUS			X	
CFSTRUCT	X	X	X	X
CHANNEL	X	X	X	X
CHSTATUS			X	
NAMELIST	X	X	X	X
PROCESS	X	X	X	X
QALIAS	M	M	M	M
QCLUSTER			M	
QLOCAL	M	M	M	M
QMGR	X		X	
QMODEL	M	M	M	M
QREMOTE	M	M	M	M
QUEUE	P	P	X	P
QSTATUS			X	
STGCLASS	X	X	X	X

Key to table symbols:

- M = MQSC only
- P = PCF only
- X = both

There are many other MQSC and PCF commands which allow you to manage other WebSphere MQ resources, and carry out other actions in addition to those summarized in Table 10 on page 72.

Table 11 shows every MQSC command, and where each command can be issued from:

- CSQINP1 initialization input data set
- CSQINP2 initialization input data set
- z/OS console (or equivalent)
- SYSTEM.COMMAND.INPUT queue and command server (from applications, CSQUTIL, or the CSQINPX initialization input data set)

Table 11. Sources from which to run MQSC commands

Command	CSQINP1	CSQINP2	z/OS console	Command input queue and server
ALTER AUTHINFO		X	X	X
ALTER BUFFPOOL			X	X
ALTER CFSTRUCT		X	X	X
ALTER CHANNEL		X	X	X
ALTER NAMELIST		X	X	X
ALTER PSID			X	X
ALTER PROCESS		X	X	X
ALTER QALIAS		X	X	X
ALTER QLOCAL		X	X	X
ALTER QMGR		X	X	X
ALTER QMODEL		X	X	X
ALTER QREMOTE		X	X	X
ALTER SECURITY	X	X	X	X
ALTER STGCLASS		X	X	X
ALTER SUB		X	X	X
ALTER TOPIC		X	X	X
ALTER TRACE	X	X	X	X
ARCHIVE LOG	X	X	X	X
BACKUP CFSTRUCT			X	X
CLEAR QLOCAL		X	X	X
DEFINE AUTHINFO		X	X	X
DEFINE BUFFPOOL	X			
DEFINE CFSTRUCT		X	X	X
DEFINE CHANNEL		X	X	X
DEFINE LOG			X	X
DEFINE NAMELIST		X	X	X
DEFINE PROCESS		X	X	X
DEFINE PSID	X		X	X
DEFINE QALIAS		X	X	X

Table 11. Sources from which to run MQSC commands (continued)

Command	CSQINP1	CSQINP2	z/OS console	Command input queue and server
DEFINE QLOCAL		X	X	X
DEFINE QMODEL		X	X	X
DEFINE QREMOTE		X	X	X
DEFINE STGCLASS		X	X	X
DEFINE SUB		X	X	X
DEFINE TOPIC		X	X	X
DELETE AUTHINFO		X	X	X
DELETE BUFFPOOL			X	X
DELETE CFSTRUCT		X	X	X
DELETE CHANNEL			X	X
DELETE NAMELIST		X	X	X
DELETE PROCESS		X	X	X
DELETE PSID			X	X
DELETE QALIAS		X	X	X
DELETE QLOCAL		X	X	X
DELETE QMODEL		X	X	X
DELETE QREMOTE		X	X	X
DELETE STGCLASS		X	X	X
DELETE SUB		X	X	X
DELETE TOPIC		X	X	X
DISPLAY ARCHIVE	X	X	X	X
DISPLAY AUTHINFO		X	X	X
DISPLAY CFSTATUS			X	X
DISPLAY CFSTRUCT		X	X	X
DISPLAY CHANNEL		X	X	X
DISPLAY CHSTATUS			X	X
DISPLAY CLUSQMGR			X	X
DISPLAY CMDSERV	X	X	X	X
DISPLAY CONN		X	X	X
DISPLAY CHINIT		X	X	X
DISPLAY GROUP		X	X	X
DISPLAY LOG	X	X	X	X
DISPLAY NAMELIST		X	X	X
DISPLAY PROCESS		X	X	X
DISPLAY QALIAS		X	X	X
DISPLAY QCLUSTER		X	X	X
DISPLAY QLOCAL		X	X	X
DISPLAY QMGR		X	X	X

Table 11. Sources from which to run MQSC commands (continued)

Command	CSQINP1	CSQINP2	z/OS console	Command input queue and server
DISPLAY QMODEL		X	X	X
DISPLAY QREMOTE		X	X	X
DISPLAY QSTATUS		X	X	X
DISPLAY QUEUE		X	X	X
DISPLAY SECURITY	X	X	X	X
DISPLAY STGCLASS		X	X	X
DISPLAY SUB		X	X	X
DISPLAY TOPIC		X	X	X
DISPLAY SYSTEM	X	X	X	X
DISPLAY THREAD		X	X	X
DISPLAY TRACE	X	X	X	X
DISPLAY USAGE		X	X	X
MOVE QLOCAL		X	X	X
PING CHANNEL			X	X
RECOVER BSDS	X	X	X	X
RECOVER CFSTRUCT			X	X
REFRESH CLUSTER		X	X	X
REFRESH QMGR		X	X	X
REFRESH SECURITY		X	X	X
RESET CHANNEL			X	X
RESET CLUSTER		X	X	X
RESET QSTATS		X	X	X
RESET TPIPE			X	X
RESOLVE CHANNEL			X	X
RESOLVE INDOUBT		X	X	X
RESUME QMGR			X	X
RVERIFY SECURITY		X	X	X
SET ARCHIVE	X	X	X	X
SET LOG	X	X	X	X
SET SYSTEM	X	X	X	X
START CHANNEL			X	X
START CHINIT		X	X	X
START CMDSERV	X	X	X	
START LISTENER			X	X
START QMGR			X	
START TRACE	X	X	X	X
STOP CHANNEL			X	X
STOP CHINIT			X	X

Table 11. Sources from which to run MQSC commands (continued)

Command	CSQINP1	CSQINP2	z/OS console	Command input queue and server
STOP CMDSERV	X	X	X	
STOP LISTENER			X	X
STOP QMGR			X	X
STOP TRACE	X	X	X	X
SUSPEND QMGR			X	X

In the *WebSphere MQ Script (MQSC) Command Reference*, each command description identifies the sources from which that command can be run.

Initialization commands

Commands in the initialization input data sets are processed when WebSphere MQ is initialized on queue manager startup. Three types of command can be issued from the initialization input data sets:

- Commands to define WebSphere MQ entities that cannot be recovered (DEFINE BUFFPOOL and DEFINE PSID for example).

  These commands must reside in the data set identified by the DDname CSQINP1. They are processed before the restart phase of initialization. They cannot be issued through the console, operations and control panels, or an application program. The responses to these commands are written to the sequential data set that you refer to in the CSQOUT1 statement of the started task procedure.
- Commands to define WebSphere MQ objects that are recoverable after restart. These definitions must be specified in the data set identified by the DDname CSQINP2. They are stored in page set zero. CSQINP2 is processed after the restart phase of initialization. The responses to these commands are written to the sequential data set that you refer to in the CSQOUT2 statement of the started task procedure.
- Commands to manipulate WebSphere MQ objects. These commands must also be specified in the data set identified by the DDname CSQINP2. For example, the WebSphere MQ-supplied sample contains an ALTER QMGR command to specify a dead-letter queue for the subsystem. The response to these commands is written to the CSQOUT2 output data set.

Note: If WebSphere MQ objects are defined in CSQINP2, WebSphere MQ attempts to redefine them each time the queue manager is started. If the queues already exist, the attempt to define them fails. If you need to define your objects in CSQINP2, you can avoid this problem by using the REPLACE parameter of the DEFINE commands, however, this will override any changes that were made during the previous run of the queue manager.

Sample initialization data set members are supplied with WebSphere MQ for z/OS. They are described in "Sample definitions supplied with WebSphere MQ" on page 40.

Initialization commands for distributed queuing:

You can also use the CSQINP2 initialization data set for the START CHINIT command. If you need a series of other commands to define your distributed

queuing environment (for example, starting listeners), WebSphere MQ provides a third initialization input data set, called CSQINPX, that is processed as part of the channel initiator started task procedure.

The MQSC commands contained in the data set are executed at the end of channel initiator initialization, and output is written to the data set specified by the CSQOUTX DD statement. You might use the CSQINPX initialization data set to start listeners for example.

A sample channel initiator initialization data set member is supplied with WebSphere MQ for z/OS. It is described in "Sample definitions supplied with WebSphere MQ" on page 40.

The WebSphere MQ for z/OS utilities

WebSphere MQ for z/OS supplies a set of utility programs to help you perform various administrative tasks, including the following:

- Perform backup, restoration, and reorganization tasks.
- Issue commands and process object definitions.
- Generate data-conversion exits.
- Modify the bootstrap data set.
- List information about the logs.
- Print the logs.
- Set up DB2 tables and other DB2 utilities.
- Process messages on the dead-letter queue.

The CSQUTIL utility

The CSQUTIL utility program is provided with WebSphere MQ for z/OS to help you perform backup, restoration, and reorganization tasks, and to issue commands and process object definitions. Through this utility program, you can invoke the following functions:

COMMAND
To issue MQSC commands, to record object definitions, and to make client-channel definition files.

COPY To read the contents of a named WebSphere MQ for z/OS message queue or the contents of all the queues of a named page set, and put them into a sequential file and retain the original queue.

COPYPAGE
To copy whole page sets to larger page sets.

EMPTY
To delete the contents of a named WebSphere MQ for z/OS message queue or the contents of all the queues of a named page set, retaining the definitions of the queues.

FORMAT
To format WebSphere MQ for z/OS page sets.

LOAD
To restore the contents of a named WebSphere MQ for z/OS message queue or the contents of all the queues of a named page set from a sequential file created by the COPY function.

PAGEINFO
To extract page set information from one or more page sets.

RESETPAGE
To copy whole page sets to other page set data sets and reset the log information in the copy.

SCOPY
To copy the contents of a queue to a data set while the queue manager is offline.

SDEFS
To produce a set of define commands for objects while the queue manager is offline.

XPARM
To convert a channel initiator parameter load module into queue manager attributes (for migration purposes).

The data conversion exit utility

The WebSphere MQ for z/OS data conversion exit utility (CSQUCVX) runs as a stand-alone utility to create data conversion exit routines.

The change log inventory utility

The WebSphere MQ for z/OS change log inventory utility program (CSQJU003) runs as a stand-alone utility to change the bootstrap data set (BSDS). You can use the utility to perform the following functions:

- Add or delete active or archive log data sets.
- Supply passwords for archive logs.

The print log map utility

The WebSphere MQ for z/OS print log map utility program (CSQJU004) runs as a stand-alone utility to list the following information:

- Log data set name and log RBA association for both copies of all active and archive log data sets. If dual logging is not active, there is only one copy of the data sets.
- Active log data sets available for new log data.
- Contents of the queue of checkpoint records in the bootstrap data set (BSDS).
- Contents of the archive log command history record.
- System and utility time stamps.

The log print utility

The log print utility program (CSQ1LOGP) is run as a stand-alone utility. You can run the utility specifying:

- A bootstrap data set (BSDS)
- Active logs (with no BSDS)
- Archive logs (with no BSDS)

The queue-sharing group utility

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

The active log preformat utility

The active log preformat utility (CSQJUFMT) formats active log data sets before they are used by a queue manager. If the active log data sets are preformatted by the utility, log write performance is improved on the queue manager's first pass through the active logs.

The dead-letter queue handler utility

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

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 12. Where to find more information about creating and managing objects

Topic	Where to look
Using initialization input data sets System parameter macros Installation verification program	WebSphere MQ for z/OS System Setup Guide
Writing administration programs Operations and control panels Utility programs	WebSphere MQ for z/OS System Administration Guide
Queue indexes MQSC commands	WebSphere MQ Script (MQSC) Command Reference
PCF commands	WebSphere MQ Programmable Command Formats and Administration Interface
WebSphere MQ clusters	WebSphere MQ Queue Manager Clusters
WebSphere MQ events	Monitoring WebSphere MQ

Monitoring and statistics

WebSphere MQ supplies facilities for monitoring the system and collecting statistics. These are discussed in the following sections:

Online monitoring

WebSphere MQ includes the following commands for monitoring the status of WebSphere MQ objects:

- DISPLAY CHSTATUS displays the status of a specified channel.

- DISPLAY QSTATUS displays the status of a specified queue.
- DISPLAY CONN displays the status of a specified connection.

For more information about these commands, see the *WebSphere MQ Script (MQSC) Command Reference*.

WebSphere MQ trace

WebSphere MQ supplies a trace facility that you can use to gather the following information while the queue manager is running:

Performance statistics

The statistics trace gathers the following information to help you monitor performance and tune your system:

- Counts of different MQI requests (message manager statistics)
- Counts of different object requests (data manager statistics)
- Information about DB2 usage (DB2 manager statistics)
- Information about Coupling Facility usage (Coupling Facility manager statistics)
- Information about buffer pool usage (buffer manager statistics)
- Information about logging (log manager statistics)
- Information about storage usage (storage manager statistics)
- Information about lock requests (lock manager statistics)

Accounting data

- The accounting trace gathers information about the CPU time spent processing MQI calls and about the number of **MQPUT** and **MQGET** requests made by a particular user.
- WebSphere MQ can also gather information about each task using WebSphere MQ. This data is gathered as a thread-level accounting record. For each thread, WebSphere MQ also gathers information about each queue used by that thread.

The data generated by the trace is sent to the System Management Facility (SMF) or the generalized trace facility (GTF).

Events

WebSphere MQ events provide information about errors, warnings, and other significant occurrences in a queue manager. By incorporating these events into your own system management application, you can monitor the activities across many queue managers, for multiple WebSphere MQ applications. In particular, you can monitor all the queue managers in your system from a single queue manager.

Events can be reported through a user-written reporting mechanism to an administration application that supports the presentation of the events to an operator. Events also enable applications acting as agents for other administration networks, for example NetView®, to monitor reports and create the appropriate alerts.

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 13. Where to find more information about monitoring and statistics

Topic	**Where to look**
WebSphere MQ trace	WebSphere MQ for z/OS System Setup Guide
Trace commands	WebSphere MQ Script (MQSC) Command Reference
WebSphere MQ events	Monitoring WebSphere MQ

Chapter 3. WebSphere MQ and other products

WebSphere MQ and CICS

This collection of topics discusses how WebSphere MQ works with CICS, using the CICS adapter and CICS bridge.

The minimum level of CICS required is specified in "Software requirements" on page 137.

The CICS adapter allows you to connect your queue manager to CICS, and enables CICS applications to use the MQI.

The optional additional WebSphere MQ CICS bridge enables applications to run a CICS program or transaction that does not use the MQI. This means that you can use your legacy applications with WebSphere MQ, without the need to rewrite them.

These topics are described in the following sections:

- "The CICS adapter"
- "The CICS bridge" on page 88
- "Where to find more information" on page 93

The CICS adapter

The CICS adapter connects a CICS subsystem to a queue manager, enabling CICS application programs to use the MQI.

You can start and stop CICS and WebSphere MQ independently, and you can establish or terminate a connection between them at any time. You can also allow CICS to connect to WebSphere MQ automatically.

The CICS adapter provides two main facilities:

- A set of control functions for use by system programmers and administrators to manage the adapter.
- MQI support for CICS applications.

In a CICS multiregion operation or intersystem communication (ISC) environment, each CICS address space can have its own attachment to the queue manager subsystem. A single CICS address space can connect to only one queue manager at a time. However, multiple CICS address spaces can connect to the same queue manager.

You can use WebSphere MQ with the CICS Extended Recovery Facility (XRF) to aid recovery from a CICS error.

The CICS adapter is supplied with WebSphere MQ and runs as a CICS External Resource Manager. WebSphere MQ also provides CICS transactions to manage the interface.

The CICS adapter uses standard CICS command-level services where required, for example, EXEC CICS WAIT and EXEC CICS ABEND. A portion of the CICS adapter runs under the control of the transaction issuing the messaging requests. Therefore, these calls for CICS services appear to be issued by the transaction.

Control functions

The CICS adapter's control functions (the CKQC transaction) let you manage the connections between CICS and WebSphere MQ dynamically. You can invoke these functions using the CICS adapter panels, from the command line, or from a CICS application. You can use the adapter's control function to:

- Start or stop a connection to a queue manager.
- Modify the current connection. For example, you can reset the connection statistics, change the adapter's trace ID number, and enable or disable the API-crossing exit.
- Display the current status of a connection and the statistics associated with that connection.
- Start or stop an instance of the task initiator transaction, CKTI. ("Task initiator transaction" is CICS terminology; in WebSphere MQ terminology, this is a trigger monitor.)
- Display details of the current instances of CKTI.
- Display details of the CICS tasks currently using the adapter.

These functions and the different methods of invoking them are described in the WebSphere MQ for z/OS System Administration Guide.

MQI support

The CICS adapter implements the MQI for use by CICS application programs. The MQI calls, and how they are used, are described in the WebSphere MQ Application Programming Guide. The adapter also supports an *API-crossing exit*, (see "The API-crossing exit" on page 87), and a trace facility.

You must linkedit all application programs that run under CICS with the supplied *API stub program* called CSQCSTUB if they are to access WebSphere MQ, unless the program is using dynamic calls. This stub provides the application with access to all MQI calls. (For information about calling the CICS stub dynamically, see the WebSphere MQ Application Programming Guide.)

For transaction integrity, the adapter supports syncpointing under the control of the CICS syncpoint manager, so that units of work can be committed or backed out as required. The adapter also supports security checking of WebSphere MQ resources when used with an appropriate security management product, such as Security Server (previously known as RACF). The adapter provides high availability with automatic reconnection after a queue manager termination, and automatic resource resynchronization after a restart. It also features an alert monitor that responds to unscheduled events such as a shut down of the queue manager.

Adapter components

Figure 15 on page 85 shows the relationship between CICS, the CICS adapter, and WebSphere MQ. CICS and the adapter share the same address space; the queue manager executes in its own address space.

Part of the adapter is a CICS task-related user exit that communicates with the WebSphere MQ message manager. CICS management modules call the exit directly; application programs call it through the supplied API stub program (CSQCSTUB). Task-related user exits and stub programs are described in the *CICS Customization Guide*.

Each CKTI transaction is normally in an **MQGET WAIT** state, ready to respond to any trigger messages that are placed on its initiation queue.

The adapter management interface provides the operation and control functions described in the WebSphere MQ for z/OS System Administration Guide.

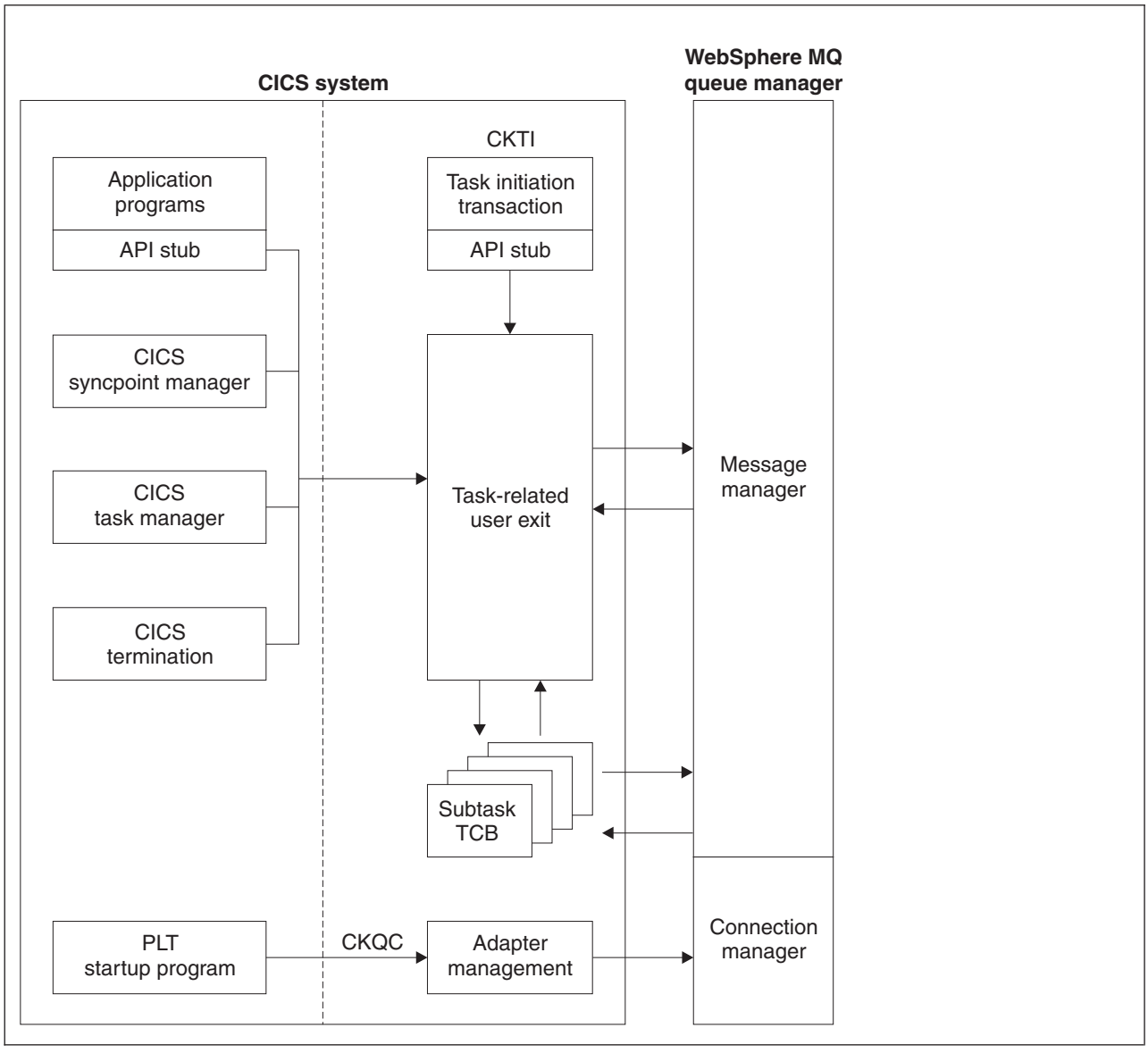


Figure 15. How CICS, the CICS adapter, and a WebSphere MQ queue manager are related

Alert monitor

The *alert monitor* transaction, CKAM, handles unscheduled events—known as *pending events*—that occur as a result of connect requests to instances of WebSphere MQ. The alert monitor generates messages that are sent to the system console.

There are two kinds of pending events:

1. **Deferred connection**

   If CICS tries to connect to WebSphere MQ before the queue manager is started, a *pending event* called a *deferred connection* is activated. When the queue manager is started, a connection request is issued by the CICS adapter, a connection is made, and the pending event is canceled.

   There can be multiple deferred connections, one of which will be connected when the queue manager is started.

2. **Termination notification**

   When a connection is successfully made to WebSphere MQ, a pending event called *termination notification* is created. This pending event expires when:

   - The queue manager shuts down normally with MODE(QUIESCE). The alert monitor issues a quiesce request on the connection.
   - The queue manager shuts down with MODE(FORCE) or terminates abnormally. After an abnormal termination, the CICS adapter waits for ten seconds and then tries a connect call. This enables the CICS system to be automatically reconnected to the queue manager when the latter is restarted.
   - The connection is shut down from the CKQC transaction.

The maximum number of pending events that CICScan handle is 99. If this limit is reached, no more events can be created until at least one current event expires.

The alert monitor terminates itself when all pending events have expired. It is subsequently restarted automatically by any new connect request.

Auto-reconnect

When CICS is connected to WebSphere MQ and the queue manager terminates, the CICS adapter tries to issue a connect request ten seconds after the stoppage has been detected. This request uses the same connect parameters that were used in the previous connect request. If the queue manager has not been restarted within the ten seconds, the connect request is deferred until the queue manager is restarted later.

Task initiator

CKTI is a WebSphere MQ-supplied CICS transaction that starts a CICS transaction when a WebSphere MQ trigger message is read, for example when a message is put onto a specific queue.

When a message is put onto an application message queue, a trigger is generated if the trigger conditions are met. The queue manager then writes a message, containing user-defined data, known as a *trigger message*, to the initiation queue that has been specified for that message queue. In a CICS environment, you can set up an instance of CKTI to monitor an initiation queue and to retrieve the trigger messages from it as they arrive. CKTI starts another CICS transaction, (specified using the DEFINE PROCESS command), which typically reads the message from the application message queue and then processes it. The process must be named on the application queue definition, not the initiation queue.

Each copy of CKTI services a single initiation queue. To start or stop a copy of CKTI, you must supply the name of the queue that this CKTI is to serve, or is

serving. You cannot start more than one instance of CKTI against the same initiation queue from a single CICS subsystem.

At CICS system initialization or at connect time, you can define a default initiation queue. If you issue a STARTCKTI or a STOPCKTI command without specifying an initiation queue, these commands are automatically interpreted as referring to the default initiation queue.

Multitasking

When WebSphere MQ is connected to CICS Transaction Server V3.1 or earlier versions, the following information applies:

- The CICS adapter optimizes the performance of a CICS to WebSphere MQ connection by exploiting multiprocessors and by removing work from the main CICS task control block (TCB), allowing multiple MQI calls to be handled concurrently.
- The adapter enables some MQI calls to be executed under subtasks, rather than under the main CICS TCB that runs the application code. All the CICS adapter administration code, including connection and disconnection from WebSphere MQ, runs under the main CICS TCB.
- The adapter attaches eight z/OS subtasks (TCBs) to be used by this CICS system. *You cannot modify this number*. Each subtask makes a connect call to WebSphere MQ. Each CICS system connected takes up nine of the connections specified on the CTHREAD system parameter. This means that you must increase the value specified for CTHREAD by nine for each CICS system connected. MQI calls can flow over those connections. When the main connection is terminated, the subtasks are disconnected and terminated automatically.

When WebSphere MQ is connected to CICS Transaction Server V3.2, the adapter does not attach subtasks but exploits the CICS Open Transaction environment and runs on CICS L8 TCBs.

The API-crossing exit

WebSphere MQ provides an API-crossing exit for use with the CICS adapter; it runs in the CICS address space. You can use this exit to intercept MQI calls as they are being run, for monitoring, testing, maintenance, or security purposes.

Using the API-crossing exit degrades WebSphere MQ performance. You should plan your use of it carefully.

CICS adapter conventions

There are a number of conventions that must be observed in applications using the adapter.

Temporary storage queue names:

The CICS adapter display function uses two temporary storage queues (MAIN) for each invoking task to store the output data for browsing. The names of the queues are *tttt*CKRT and *tttt*CKDP, where *tttt* is the terminal identifier of the terminal from which the display function is requested.

Do not try to access these queues.

MQGET:

When the CICS adapter puts a task on a CICS wait because the WAIT option was used with the **MQGET** call and there was no message available, the RESOURCE NAME used is GETWAIT and the RESOURCE_TYPE is MQSeries®.

When the CICS adapter puts a task on a CICS wait because of a need to perform task switching the RESOURCE NAME used is TASKSWCH and the RESOURCE_TYPE is MQSeries.

ENQUEUE names:

The CICS adapter uses the name: `CSQ.genericapplid(8).QMGR` to issue CICS ENQ and CICS DEQ calls during processing, for example, starting and stopping the connection. You should not use similar names for CICS ENQ or DEQ purposes.

The CICS bridge

The WebSphere MQ-CICS bridge is the component of WebSphere MQ for z/OS that allows direct access from WebSphere MQ applications to applications on your CICS system. In bridge applications there are no WebSphere MQ calls within the CICS application (the bridge enables *implicit MQI support*). This means that you can re-engineer legacy applications that were controlled by 3270-connected terminals to be controlled by WebSphere MQ messages, without having to rewrite, recompile, or re-link them.

WebSphere MQ applications use the CICS header (the MQCIH structure) in the message data to ensure that the applications can execute as they did when driven by nonprogrammable terminals.

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

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

A *transaction* is a CICS transaction designed to run on a 3270 terminal. This transaction can use BMS or TC commands. It can be conversational or part of a pseudoconversation. It is permitted to issue syncpoints. For information about the transactions that can be run, see the *CICS External Interfaces Guide*.

When to use the CICS bridge

The CICS bridge allows an application to run a single CICS program or a 'set' of CICS programs (often referred to as a unit of work). It caters for the application that waits for a response to come back before it runs the next CICS program (synchronous processing) and for the application that requests one or more CICS programs to run, but doesn't wait for a response (asynchronous processing).

The CICS bridge also allows an application to run a 3270-based CICS transaction, without knowledge of the 3270 data stream.

The CICS bridge uses standard CICS and WebSphere MQ security features and you can configure it to authenticate, trust, or ignore the requestor's user ID.

Given this flexibility, there are many instances where you can use the CICS bridge. For example, when you want to do any of the following:

- Write a new WebSphere MQ application that needs access to logic or data (or both) that reside on your CICS server.
- Run CICS programs from a Lotus Notes® application.
- Access your CICS applications from your WebSphere MQ Classes for Java™ client application

System configuration for the CICS bridge:

When you are setting your system up, you should ensure that:

- Both WebSphere MQ and CICS are running in the same z/OS image.
- The WebSphere MQ request queue is local to the CICS bridge, however the response queue can be local or remote.
- The CICS bridge tasks normally run in the same CICS as the bridge monitor. The user programs can be in the same or a different CICS system.
  - CICS transaction routing can be used, if the CICS system is CICS Transaction Server V2.2 or higher.
- The WebSphere MQ-CICS adapter is enabled.
- A specific CICS system can be referenced in the MQCIH header.

When using shared queues, it is possible to use multiple bridge monitors on the CICS bridge in multiple CICS regions, however all bridge monitors must be associated with WebSphere MQ queue managers.

Running CICS DPL programs

Data necessary to run the program is provided in the WebSphere MQ message. The bridge builds a COMMAREA from this data, and runs the program using EXEC CICS LINK. Figure 16 on page 90 shows the sequence of actions taken to process a single message to run a CICS DPL program:

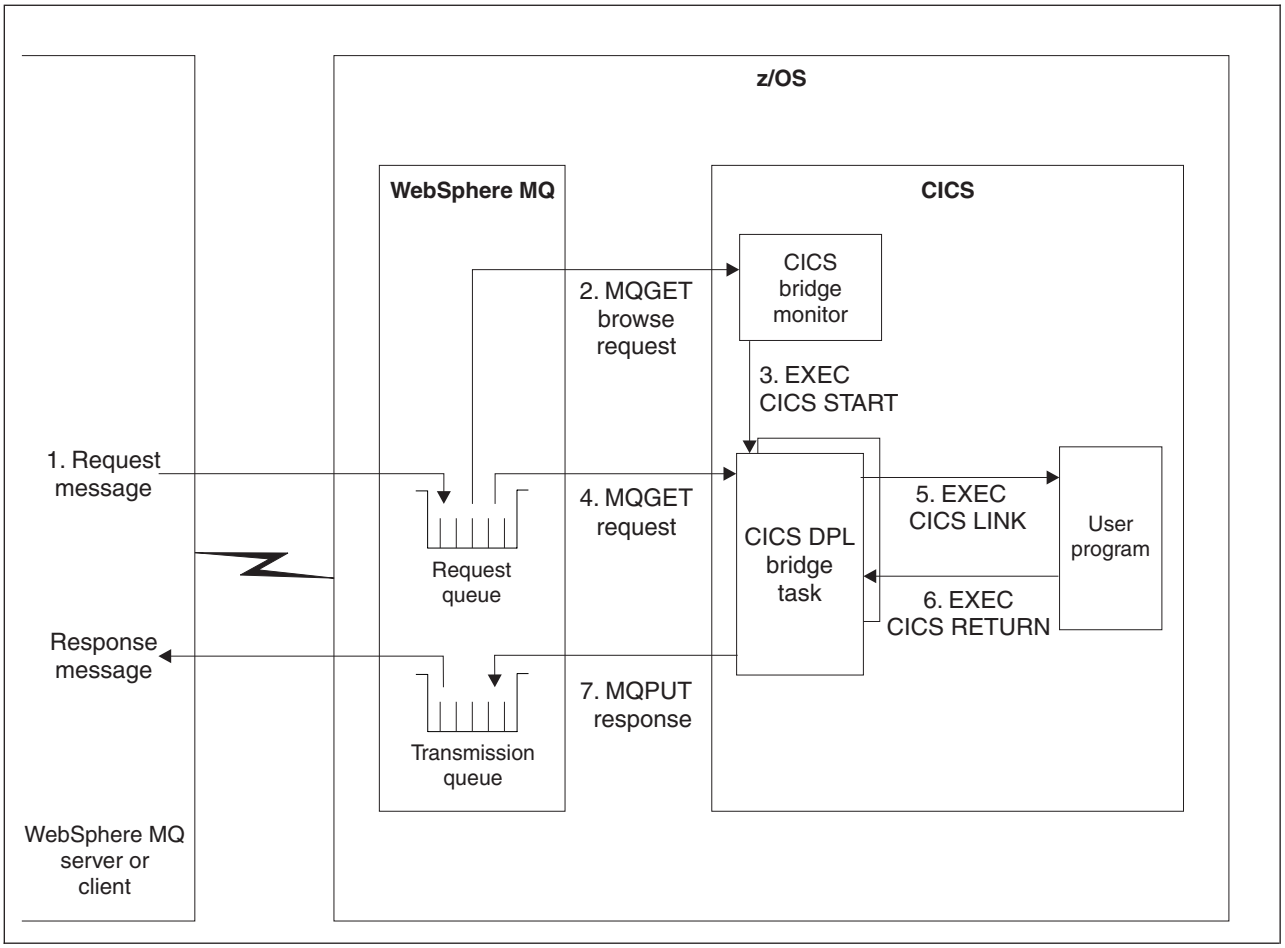


Figure 16. Components and data flow to run a CICS DPL program

The following takes each step in turn, and explains what takes place:

1. A message, with a request to run a CICS program, is put on the request queue.
2. The CICS bridge monitor task, which is constantly browsing the queue, recognizes that a 'start unit of work' message is waiting (*CorrelId*=MQCI_NEW_SESSION).
3. Relevant authentication checks are made, and a CICS DPL bridge task is started with the appropriate authority, with a particular userid (depending on the options used to start the bridge monitor).
4. The CICS DPL bridge task removes the message from the request queue.
5. The CICS DPL bridge task builds a COMMAREA from the data in the message and issues an EXEC CICS LINK for the program requested in the message.
6. The program returns the response in the COMMAREA used by the request.
7. The CICS DPL bridge task reads the COMMAREA, creates a message, and puts it on the reply-to queue specified in the request message. All response messages (normal and error, requests and replies) are put to the reply-to queue with default context.
8. The CICS DPL bridge task ends. If this is the last flow in the transaction then the transaction ends, if it is not the last message, the transaction waits until the next message is received or the specified timeout interval expires.

A unit of work can be just a single user program, or it can be multiple user programs. There is no limit to the number of messages you can send to make up a unit of work.

In this scenario, a unit of work made up of many messages works in the same way, with the exception that the CICS bridge task waits for the next request message in the final step unless it is the last message in the unit of work.

Running CICS 3270 transactions

Data necessary to run the transaction is provided in the WebSphere MQ message. The CICS transaction runs as if it has a real 3270 terminal, but instead uses one or more MQ messages to communicate between the CICS transaction and the WebSphere MQ application

Unlike traditional 3270 emulators, the bridge does not work by replacing the VTAM flows with WebSphere MQ messages. Instead, the message consists of a number of parts called vectors, each of which corresponds to an EXEC CICS request. Therefore the application is talking directly to the CICS transaction, rather than through an emulator, using the actual data used by the transaction (known as application data structures or ADSs).

Figure 17 on page 92 shows the sequence of actions taken to process a single message to run a CICS 3270 transaction.

The following takes each step in turn, and explains what takes place:

1. A message, with a request to run a CICS transaction, is put on the request queue.
2. The CICS bridge monitor task, which is constantly browsing the queue, recognizes that a 'start unit of work' message is waiting (*CorrelId*=MQCI_NEW_SESSION).
3. Relevant authentication checks are made, and a CICS 3270 bridge task is started with the appropriate authority, with a particular userid (depending on the options used to start the bridge monitor).
4. The WebSphere MQ-CICS bridge removes the message from the queue and changes task to run a user transaction
5. Vectors in the message provide data to answer all terminal-related input EXEC CICS requests in the transaction.
6. Terminal-related output EXEC CICS requests result in output vectors being built.
7. The WebSphere MQ-CICS bridge builds all the output vectors into a single message and puts this on the reply-to queue.
8. The CICS 3270 bridge task ends. If this is the last flow in the transaction then the transaction ends, if it is not the last message, the transaction waits until the next message is received or the specified timeout interval expires.

Note: The WebSphere MQ CICS bridge is a WebSphere MQ supplied CICS exit associated with the bridge transaction.

For CICS Transaction Server 2.2 and later this uses the CICS Link3270 mechanism. For earlier releases it uses the CICS START BREXIT mechanism. The MQ-CICS 3270 bridge is not supported on CICS Transaction Server V1.2 and earlier.

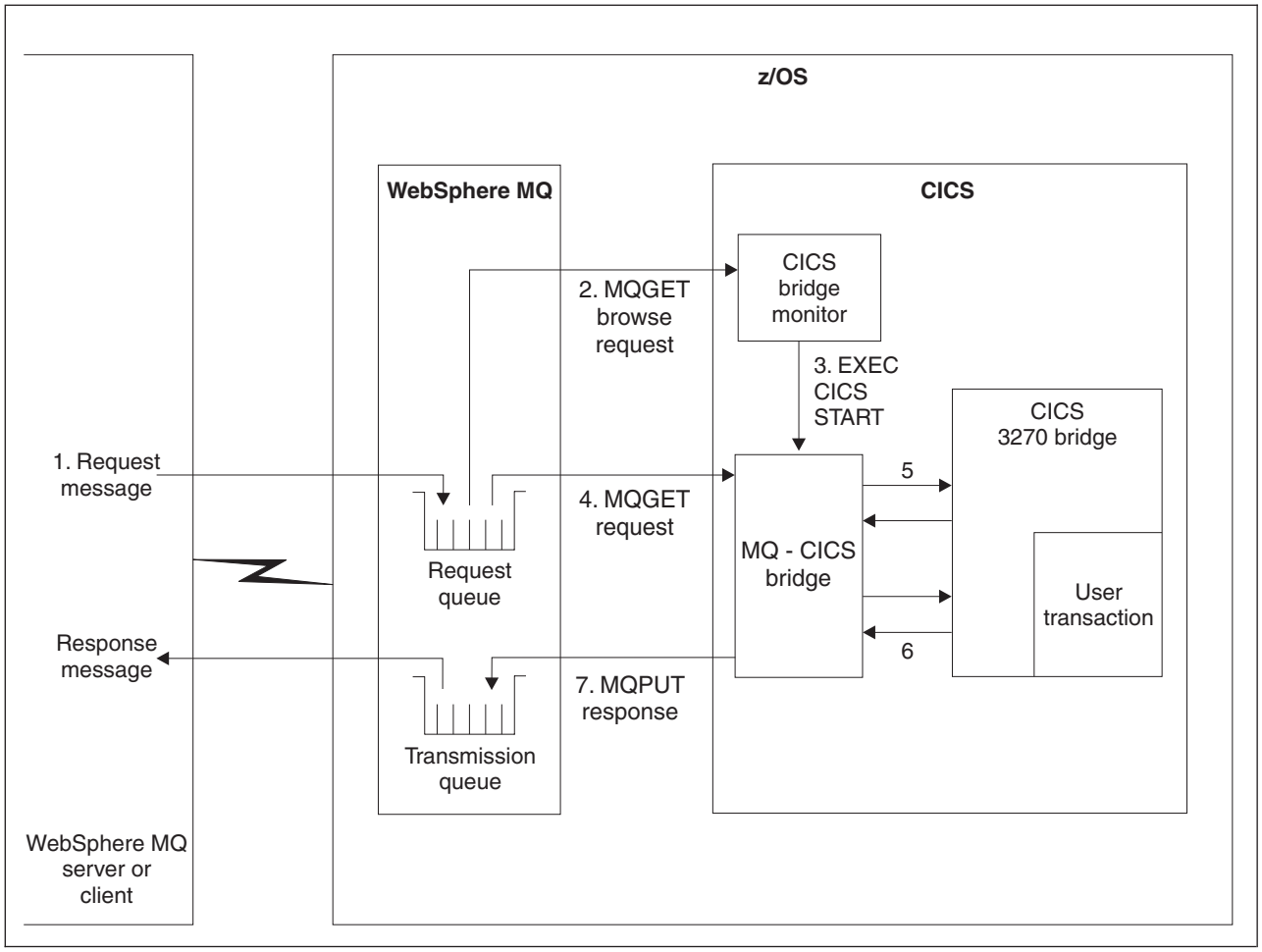


Figure 17. Components and data flow to run a CICS 3270 transaction

A traditional CICS application usually consists of one or more transactions linked together as a pseudoconversation. In general, each transaction is started by the 3270 terminal user entering data onto the screen and pressing an AID key. This model of application can be emulated by a WebSphere MQ application. A message is built for the first transaction, containing information about the transaction, and input vectors. This is put on the queue. The reply message consists of the output vectors, the name of the next transaction to be run, and a token that is used to represent the pseudoconversation. The WebSphere MQ application builds a new input message, with the transaction name set to the next transaction and the facility token and remote system id set to the value returned on the previous message. Vectors for this second transaction are added to the message, and the message put on the queue. This process is continued until the application ends.

When using CICS releases prior to CICS TS 2.2 the first message for each transaction must be a new session message. When using CICS TS 2.2 or later, it is possible to include all of the WebSphere MQ messages for multiple transactions within the same bridge session which reduces monitoring overheads and improves performance.

An alternative approach to writing CICS applications is the conversational model. In this model, the original message might not contain all the data to run the transaction. If the transaction issues a request that cannot be answered by any of the vectors in the message, a message is put onto the reply-to queue requesting

more data. The WebSphere MQ application gets this message and puts a new message back to the queue with a vector to satisfy the request.

For more information, see the *WebSphere MQ Application Programming Guide*, the *WebSphere MQ Application Programming Reference*, and the *CICS Internet and External Interfaces Guide*.

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 14. Where to find more information about using CICS with WebSphere MQ

Topic	Where to look
System parameters Setting up the CICS adapter Setting up the CICS bridge	WebSphere MQ for z/OS System Setup Guide
Operating the CICS adapter Operating the CICS bridge	WebSphere MQ for z/OS System Administration Guide
Console messages	WebSphere MQ for z/OS Messages and Codes
Writing CICS applications API-crossing exit	WebSphere MQ Application Programming Guide, WebSphere MQ Application Programming Reference
Writing CICS bridge applications	*CICS External Interfaces Guide*, WebSphere MQ Application Programming Guide, WebSphere MQ Application Programming Reference

WebSphere MQ and IMS

This chapter discusses how WebSphere MQ works with IMS. The IMS adapter allows you to connect your queue manager to IMS, and enables IMS applications to use the MQI.

The optional additional WebSphere MQ IMS bridge enables applications to run an IMS application that does not use the MQI. This means that you can use your legacy applications with WebSphere MQ, without the need to rewrite them.

These topics are described in the following sections:

The IMS adapter

The WebSphere MQ adapters enable different application environments to send and receive messages through a message queuing network. The IMS adapter is the interface between IMS application programs and a WebSphere MQ subsystem. It makes it possible for IMS application programs to use the MQI.

The IMS adapter receives and interprets requests for access to WebSphere MQ using the External Subsystem Attach Facility (ESAF) provided by IMS. This facility is described in the *IMS Customization Guide*. Usually, IMS connects to WebSphere MQ automatically without operator intervention.

The IMS adapter provides access to WebSphere MQ resources for programs running in the following modes or states:

- Task (TCB) mode
- Problem state
- Non-cross-memory mode
- Non-access register mode

The adapter provides a connection thread from an application task control block (TCB) to WebSphere MQ.

The adapter supports a two-phase commit protocol for changes made to resources owned by WebSphere MQ with IMS acting as the syncpoint coordinator. Conversations where IMS is not the syncpoint coordinator, for example APPC-protected (SYNCLVL=SYNCPT) conversations, are not supported by the IMS adapter.

The adapter also provides a trigger monitor transaction (CSQQTRMN). This is described in "The IMS trigger monitor."

You can use WebSphere MQ with the IMS Extended Recovery Facility (XRF) to aid recovery from a IMS error. For more information about XRF, see the *IMS Administration Guide: System* manual.

Using the adapter

The application programs and the IMS adapter run in the same address space. The queue manager is separate, in its own address space.

You must link-edit each program that issues one or more MQI calls to a suitable IMS language interface module, and, unless it uses dynamic MQI calls, the WebSphere MQ-supplied API stub program, CSQQSTUB. When the application issues an MQI call, the stub transfers control to the adapter through the IMS external subsystem interface, which manages the processing of the request by the message queue manager.

System administration and operation with IMS

An authorized IMS terminal operator can issue IMS commands to control and monitor the connection to WebSphere MQ. However, the IMS terminal operator has no control over the WebSphere MQ address space. For example, the operator cannot shut down WebSphere MQ from an IMS address space.

The IMS trigger monitor

The IMS trigger monitor (CSQQTRMN) is a WebSphere MQ-supplied IMS application that starts an IMS transaction when a WebSphere MQ event occurs, for example, when a message is put onto a specific queue.

How it works:

When a message is put onto an application message queue, a trigger is generated if the trigger conditions are met. The queue manager then writes a message (containing some user-defined data), known as a *trigger message*, to the initiation queue that has been specified for that message queue. In an IMS environment, you can start an instance of CSQQTRMN to monitor an initiation queue and to retrieve the trigger messages from it as they arrive. Typically, CSQQTRMN schedules another IMS transaction by an INSERT (ISRT) to the IMS message queue. The started IMS application reads the message from the application message queue and then processes it. CSQQTRMN must run as a non-message BMP.

Each copy of CSQQTRMN services a single initiation queue. When it has started, the trigger monitor runs until WebSphere MQ or IMS ends.

The APPLCTN macro for CSQQTRMN must specify SCHDTYP=PARALLEL.

Because the trigger monitor is a batch-oriented BMP, IMS transactions that are started by the trigger monitor contain the following:

- Blanks in the LTERM field of the IOPCB
- The PSB name of the trigger monitor BMP in the Userid field of the IOPCB

If the target IMS transaction is protected by Security Server (previously known as RACF), you might need to define CSQQTRMN as a user ID to Security Server.

The IMS bridge

The WebSphere MQ-IMS bridge is the component of WebSphere MQ for z/OS that allows direct access from WebSphere MQ applications to applications on your IMS system (the bridge enables *implicit MQI support*). This means that you can re-engineer legacy applications that were controlled by 3270-connected terminals to be controlled by WebSphere MQ messages, without having to rewrite, recompile, or re-link them. The bridge is an IMS *Open Transaction Manager Access* (OTMA) client.

In bridge applications there are no WebSphere MQ calls within the IMS application. The application gets its input using a GET UNIQUE (GU) to the IOPCB and sends its output using an ISRT to the IOPCB. WebSphere MQ applications use the IMS header (the MQIIH structure) in the message data to ensure that the applications can execute as they did when driven by nonprogrammable terminals. If you are using an IMS application that processes multi-segment messages, note that all segments should be contained within one WebSphere MQ message.

The IMS bridge is illustrated in Figure 18 on page 96.

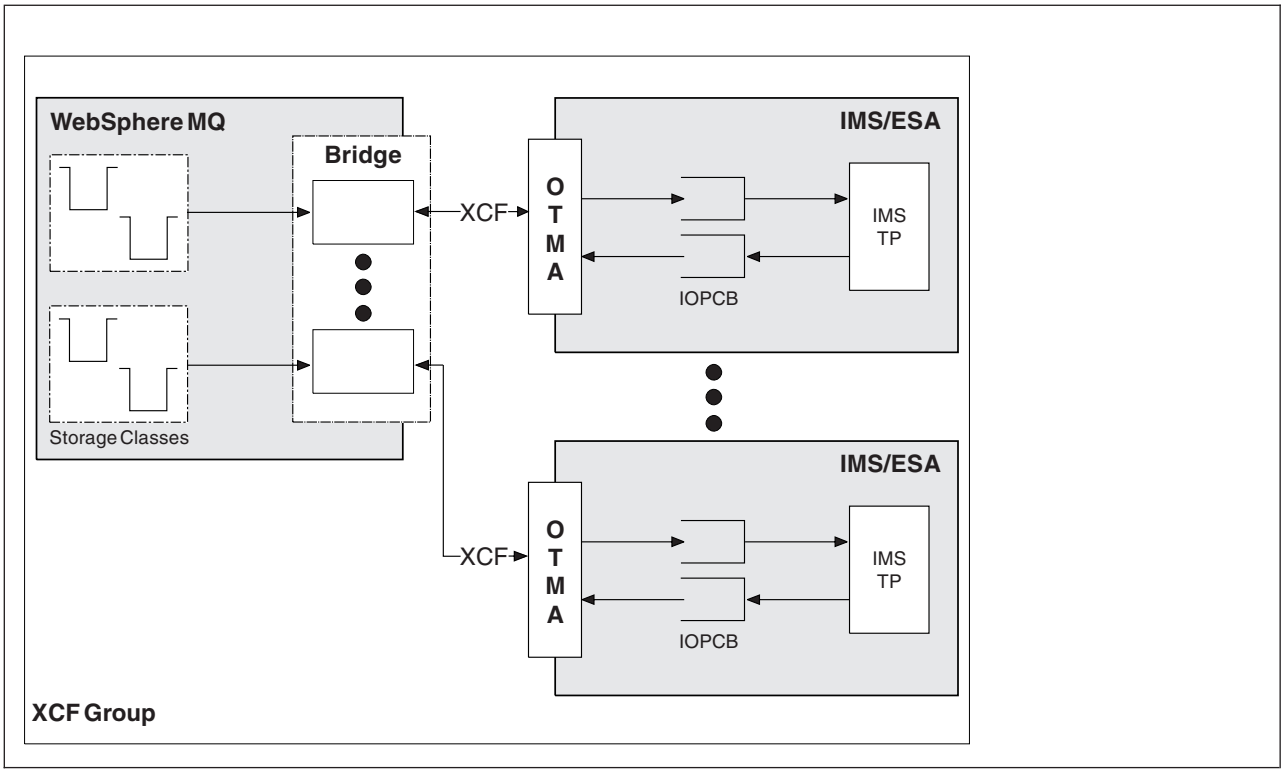


Figure 18. The WebSphere MQ-IMS bridge

A queue manager can connect to one or more IMS systems, and more than one queue manager can connect to one IMS system. The only restriction is that they must all belong to the same XCF group and must all be in the same sysplex.

What is OTMA?

The IMS OTMA facility is a transaction-based connectionless client/server protocol that runs on IMS Version 5.1 or later. It functions as an interface for host-based communications servers accessing IMS TM applications through the z/OS *Cross Systems Coupling Facility* (XCF).

OTMA enables clients to connect to IMS to provide high performance for interactions between clients and IMS for a large network or large number of sessions. OTMA is implemented in a z/OS sysplex environment. Therefore, the domain of OTMA is restricted to the domain of XCF.

Submitting IMS transactions from WebSphere MQ

To submit an IMS transaction that uses the bridge, applications put messages on a WebSphere MQ queue as usual. The messages contain IMS transaction data; they can have an IMS header (the MQIIH structure) or allow the WebSphere MQ-IMS bridge to make assumptions about the data in the message.

WebSphere MQ then puts the message to an IMS queue (it is queued in WebSphere MQ first to enable the use of syncpoints to assure data integrity). The storage class of the WebSphere MQ queue determines whether the queue is an *OTMA queue* (that is, a queue used to transmit messages to the WebSphere MQ-IMS bridge) and the particular IMS partner to which the message data is sent.

Remote queue managers can also start IMS transactions by writing to these OTMA queues on WebSphere MQ for z/OS.

Data returned from the IMS system is written directly to the WebSphere MQ reply-to queue specified in the message descriptor structure (MQMD). (This might be a transmission queue to the queue manager specified in the *ReplyToQMgr* field of the MQMD.)

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 15. Where to find more information about using IMS with WebSphere MQ

Topic	Where to look
Setting up the IMS adapter Setting up the IMS bridge	WebSphere MQ for z/OS System Setup Guide
Operating the IMS adapter Operating the IMS bridge	WebSphere MQ for z/OS System Administration Guide
Console messages	WebSphere MQ for z/OS Messages and Codes
Writing IMS applications	WebSphere MQ Application Programming Guide
IMS Open Transaction Manager Access (OTMA)	*IMS/ESA® Open Transaction Manager Access Guide*
MQIIH structure	WebSphere MQ Application Programming Reference

WebSphere MQ and z/OS Batch and TSO

This chapter discusses how WebSphere MQ works with z/OS Batch and TSO. It contains the following sections:

Introduction to the Batch adapters

The Batch/TSO adapters are the interface between z/OS application programs running under JES, TSO, or UNIX® System Services and WebSphere MQ. They enable z/OS application programs to use the MQI.

The adapters provide access to WebSphere MQ resources for programs running in the following modes or states:

- Task (TCB) mode
- Problem or supervisor state

- Non-cross-memory mode
- Non-access register mode

Connections between application programs and WebSphere MQ are at the task level. The adapters provide a connection thread from an application task control block (TCB) to WebSphere MQ.

The Batch/TSO adapter supports a single-phase commit protocol for changes made to resources owned by WebSphere MQ. It does not support multi-phase commit protocols. The RRS adapter enables WebSphere MQ applications to participate in two-phase commit protocols with other RRS-enabled products, coordinated by z/OS Resource Recovery Services (RRS).

The adapters use the z/OS STIMERM service to schedule an asynchronous event every second. This event runs an interrupt request block (IRB) that does not involve any waiting by the batch application's task. This IRB checks to see if the WebSphere MQ termination ECB has been posted. If the termination ECB has been posted, the IRB posts any application ECBs that are waiting on an event in WebSphere MQ (for example, a signal or a wait).

The Batch/TSO adapter

The WebSphere MQ Batch/TSO adapter provides WebSphere MQ support for z/OS Batch and TSO applications. All application programs that run under z/OS Batch or TSO must have the API stub program CSQBSTUB link-edited with them. The stub provides the application with access to all MQI calls. You use single-phase commit and backout for applications by issuing the MQI calls **MQCMIT** and **MQBACK**.

The RRS adapter

Resource Recovery Services (RRS) is a subcomponent of z/OS that provides a system-wide service for coordinating two-phase commit across z/OS products. The WebSphere MQ Batch/TSO RRS adapter (the RRS adapter) provides WebSphere MQ support for z/OS Batch and TSO applications that want to use these services. The RRS adapter enables WebSphere MQ to become a full participant in RRS coordination. Applications can participate in two-phase commit processing with other products that support RRS (for example, DB2).

The RRS adapter provides two stubs; you must link-edit application programs that want to use RRS with one of these stubs.

CSQBRSTB
: This stub allows you to use two-phase commit and backout for applications by using the RRS callable resource recovery services instead of the MQI calls **MQCMIT** and **MQBACK**.

  You must also link-edit module ATRSCSS from library SYS1.CSSLIB with your application. If you use the MQI calls **MQCMIT** and **MQBACK**, you will receive return code `MQRC_ENVIRONMENT_ERROR`.

CSQBRRSI
: This stub allows you to use MQI calls **MQCMIT** and **MQBACK**; WebSphere MQ actually implements these calls as the **SRRCMIT** and **SRRBACK** RRS calls.

For information about building application programs that use the RRS adapter, see the WebSphere MQ Application Programming Guide.

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 16. Where to find more information about using z/OS Batch with WebSphere MQ

Topic	Where to look
Setting up the Batch adapters	WebSphere MQ for z/OS System Setup Guide
Writing applications to use the Batch adapter	WebSphere MQ Application Programming Guide
RRS callable resource recovery services	*MVS Programming: Callable Services for High Level Languages*

WebSphere MQ and WebSphere Application Server

This chapter discusses the use of WebSphere MQ for z/OS by the WebSphere Application Server.

Applications written in Java that are running under WebSphere Application Server can use the Java Messaging Service (JMS) specification to perform messaging. Point-to-point messaging in this environment may be provided by a WebSphere MQ for z/OS queue manager.

A benefit of using a WebSphere MQ for z/OS queue manager to provide the point-to-point messaging is that connecting JMS applications can participate fully in the functionality of a WebSphere MQ network. For example, they can make use of the IMS Bridge, or exchange messages with queue managers running on other platforms.

Connection between WebSphere Application Server and a queue manager

You can choose either *client transport* or *bindings transport* for the queue connection factory object. If you choose bindings transport, the WebSphere Application Server and the queue manager must both exist on the same z/OS image. If you choose client transport, you must install the *Client Attachment* feature of WebSphere MQ for z/OS.

Both types of connection support transactional applications: the client transport by using XA protocols; the bindings transport by using a WebSphere Application Server stub, CSQBWSTB, which uses RRS services.

For more information about configuring queue connection factories see *WebSphere MQ Using Java*.

Using WebSphere MQ functions from JMS applications

By default, JMS messages held on WebSphere MQ queues use an MQRFH2 header to hold some of the JMS message header information. Many legacy WebSphere MQ applications cannot process messages with these headers, and require their own characteristic headers, for example the MQCIH for CICS Bridge, or MQWIH for WebSphere MQ Workflow applications. The section ″Mapping JMS to a native WebSphere MQ application″ in the chapter ″JMS Messages″ of *WebSphere MQ Using Java* covers these special considerations.

Chapter 4. Planning your WebSphere MQ environment

Planning your storage and performance requirements

This chapter discusses the storage and performance requirements for WebSphere MQ for z/OS. It contains the following sections:

For full information about workload management and defining goals through the service definition, see *z/OS MVS Planning: Workload Management*.

This section suggests how to set the z/OS workload management importance and velocity goals relative to other important work in your system.

Use the following service classes:

The default SYSSTC service class

- VTAM and TCP/IP address spaces
- IRLM address space (IRLMPROC)

Note: The VTAM, TCP/IP, and IRLM address spaces must have a higher dispatching priority than all of the DBMS address spaces, their attached address spaces, and their subordinate address spaces. Do not allow workload management to reduce the priority of VTAM, TCP/IP, or IRLM to (or below) that of the other DBMS address spaces

A high velocity goal and importance of 1 for a service class whose name you define, such as PRODREGN, for the following:

- MQ queue manager and channel initiator address spaces
- DB2 (all address spaces, except for the DB2-established stored procedures address space)
- CICS (all region types)
- IMS (all region types except BMPs)

A high velocity goal is good for ensuring that startups and restarts are performed as quickly as possible for all these address spaces.

The velocity goals for CICS and IMS regions are only important during startup or restart. After transactions begin running, workload management ignores the CICS or IMS velocity goals and assigns priorities based on the response time goals of the transactions that are running in the regions. These transaction goals should reflect the relative priority of the business applications they implement. They may typically have an importance value of 2. Any batch applications using MQ should similarly have velocity goals and importance reflecting the relative priority of the business applications they implement. Typically the importance and velocity goals will be less than those for PRODREGN.

Address space storage

Storage requirements can be divided into the following categories:

- Common storage
- Private storage for the queue manager
- Private storage for the channel initiator

Common storage

Each WebSphere MQ for z/OS subsystem has the following approximate storage requirements:

- CSA 4 KB
- ECSA 800 KB, plus the size of the trace table specified in the TRACTBL parameter of the CSQ6SYSP system parameter macro. This macro is described in the *WebSphere MQ for z/OS System Setup Guide*.

In addition, each concurrent WebSphere MQ logical connection requires about 5 KB of ECSA. When a task ends, other WebSphere MQ tasks can reuse this storage. WebSphere MQ does not release the storage until the queue manager is shut down, so you can calculate the maximum amount of ECSA required by multiplying the maximum number of concurrent logical connections by 5 KB. Concurrent logical connections are the number of:

- Tasks (TCBs) in Batch, TSO, USS, IMS and DB/2 SPAS regions that have connected to WebSphere MQ, but not disconnected.
- CICS transactions that have issued a WebSphere MQ request, but have not terminated
- JMS QueueSessions that have been created (for bindings connection), but not yet destroyed or garbage collected.
- Active WebSphere MQ channels.

The channel initiator typically requires ECSA usage of up to 160 KB.

Queue manager private region storage usage

Though WebSphere MQ can make use of z/OS memory objects above the 2 GB bar, substantial amounts of storage below the bar are used for certain purposes.

WebSphere MQ can make use of z/OS memory objects above the 2 GB bar for some, but not all, functions.

Within the 2 GB address space region below the bar, the largest use of private region storage is for buffer pools. Each buffer pool size is determined at queue manager initialization, and storage is allocated for them when a page set using that buffer pool is connected. The ALTER BUFFPOOL command allows the sizes of buffer pools to be dynamically changed. Other substantial uses of private storage are:

- Messages stored on indexed queues
- Open handles
- Long running units of work containing large, or many messages

Though these are not large in themselves, there are potentially many of them, or they can remain for a long time.

A periodically issued message, CSQY220I, indicates the amount of private region storage in use below the 2 GB bar, and the amount remaining.

Channel initiator private region storage usage

Every channel uses approximately 170 KB of extended private region in the channel initiator address space. Storage is increased by message size if messages larger than 32 KB are transmitted. This increased storage is freed when:

- A sending or client channel requires less than half the current buffer size for 10 consecutive messages.
- A heartbeat is sent or received.

The storage is freed for reuse within the LE environment, but is not seen as free by the z/OS virtual storage manager. This means that the upper limit for the number of channels is dependent on message size and arrival patterns as well as on limitations of individual user systems on extended private region size. The upper limit on the number of channels is likely to be approximately 9000 on many systems because the extended region size is unlikely to exceed 1.6 GB. The use of message sizes larger than 32 KB reduces the maximum number of channels in the system. For example, if messages that are 100 MB long are transmitted, and an extended region size of 1.6 GB is assumed, the maximum number of channels is 15.

Region sizes

Suggested region sizes for a queue manager and channel initiator, for a test or production system.

The following table shows suggested values for region sizes below the 2 GB bar. Two sets of values are given; one set is suitable for a test system, the other for a production system or a system that will become production eventually.

Table 17. Suggested definitions for JCL region sizes

Definition setting	Test system	Production system
Queue manager	REGION=7168K	REGION=0M
Channel initiator	REGION=7168K	REGION=0M

The region sizes suggested for the test system (REGION=7168K for both the queue manager region and channel initiator region) allow the queue manager and channel initiator to allocate 7 MB of private virtual storage below the 16 MB line (PVT) and up to 32 MB of extended private virtual storage above the 16 MB line (extended PVT).

These values are insufficient for a production system with large buffer pools and many active channels. The production region sizes chosen (REGION=0M) allow all available private storage above and below the 16 MB line to be used, although WebSphere MQ uses very little storage below the 16 MB line.

Note: You can use the z/OS exit IEALIMIT to override the region limits below the 16 MB line and IEFUSI to override the region limits above and below the 16 MB line.

In addition to the virtual storage below the 2 GB bar, the queue manager has a requirement for z/OS memory objects in storage located above the bar. Use the MEMLIMIT parameter in the JCL of the queue manager stored procedure, xxxxMSTR, to restrict the total number of virtual pages above the bar which the queue manager address space can use, for example `MEMLIMIT=2G`. Other mechanisms, for example the MEMLIMIT parm in the SMFPRMxx member of SYS1.PARMLIB or IEFUSI exit might be used at your installation to provide a

default amount of virtual storage above the bar for z/OS address spaces. See the z/OS MVS™ Extended Addressability Guide for full details of limiting storage above the bar.

Data storage

See the following sections for details of how to plan your data storage:

- **Logs and archive storage**

  "Logs and archive storage" on page 123 describes how to determine how much storage your active log and archive data sets require, depending on the volume of messages that your WebSphere MQ system handles and how often the active logs are off-loaded to your archive data sets.
- **DB2 storage**

  "DB2 storage" on page 118 describes how to determine how much storage DB2 requires for the WebSphere MQ data.
- **Coupling Facility storage**

  "Planning the size of your structures" on page 114 describes how to determine how large to make your Coupling Facility structures.
- **Page set and message storage**

  "Planning your page sets and buffer pools" on page 106 describes how to determine how much storage your page data sets require, depending on the sizes of the messages that your applications exchange, on the numbers of these messages, and on the rate at which they are created or exchanged.

Library storage

You need to allocate storage for the product libraries. The exact figures depend on your configuration, but an estimate of the space required by the distribution libraries is 80 MB. The target libraries require about 72 MB. Additionally, you require space for the SMP/E libraries.

You should refer to the program directory supplied with WebSphere MQ for z/OS for information about the required libraries and their sizes.

The thlqual.SCSQMVR1 library must be in PDS-E format.

System LX usage

Each defined WebSphere MQ subsystem reserves one system linkage index (LX) at IPL time, and a number of non-system linkage indexes when the queue manager is started. The system linkage index is reused when the queue manager is stopped and restarted. Similarly, distributed queueing reserves one non-system linkage index. In the unlikely event of your z/OS system having inadequate system LXs defined, you might need to take these reserved system LXs into account.

z/OS performance options for WebSphere MQ

With workload management, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as CPU and storage, should be given to the work to meet its goal. Workload management controls the dispatching priority based on the goals you supply. Workload management raises or lowers the priority

as needed to meet the specified goal. Thus, you do not need to fine-tune the exact priorities of every piece of work in the system and can focus instead on business objectives.

The three kinds of goals are:

Response time
How quickly you want the work to be processed

Execution velocity
How fast the work should be run when ready, without being delayed for processor, storage, I/O access, and queue delay

Discretionary
A category for low priority work for which there are no performance goals

Response time goals are generally appropriate for end user applications. For example, CICS users might set work load goals as response time goals. For WebSphere MQ address spaces, velocity goals are more appropriate. A small amount of the work done in the queue manager is counted toward this velocity goal but this work is critical for performance. Most of the work done by the queue manager counts towards the performance goal of the end user application. Most of the work done by the channel initiator address space counts towards its own velocity goal. The receiving and sending of MQ messages, which the channel initiator accomplishes, is typically very important for the performance of business applications using them.

Determining z/OS workload management importance and velocity goals

Where to find more information

You can find more information about the topics discussed in this chapter from the following sources:

Table 18. Where to find more information about storage requirements

Topic	Where to look
System parameters	WebSphere MQ for z/OS System Setup Guide
Storage required to install WebSphere MQ	*WebSphere MQ for z/OS Program Directory*
IEALIMIT and IEFUSI exits	*MVS Installation Exits*, available from the zSeries Web site http://www.ibm.com/servers/eserver/zseries/zos/bkserv/
Latest information	WebSphere MQ SupportPac Web site http://www.ibm.com/software/integration/support/supportpacs
Workload management and defining goals through the service definition	*z/OS MVS Planning: Workload Management*

Planning your page sets and buffer pools

This chapter contains the following sections:

Planning your page sets

When deciding on the most appropriate settings for page set definitions, there are a number of factors that you should consider.

Page set usage

In the case of short-lived messages, few pages are normally used on the page set and there is little or no I/O to the data sets except at startup, during a checkpoint, or at shutdown.

In the case of long-lived messages, those pages containing messages are normally written out to disk. This is performed by the queue manager in order to reduce restart time.

You should separate short-lived messages from long-lived messages by placing them on different page sets and in different buffer pools.

Number of page sets

Using several large page sets can make the role of the WebSphere MQ administrator easier because it means that you need fewer page sets, making the mapping of queues to page sets simpler.

Using multiple, smaller page sets has a number of advantages. For example, they take less time to back up, and I/O can be carried out in parallel during backup and restart. However, consider that this adds a significant overhead to the role of the WebSphere MQ administrator, who is required to map each queue to one of a much greater number of page sets.

You are recommended to define at least five page sets, as follows:

- A page set reserved for object definitions (page set zero)
- A page set for system-related messages
- A page set for performance-critical long-lived messages
- A page set for performance-critical short-lived messages
- A page set for all other messages

"Defining your buffer pools" on page 112 explains the performance advantages of distributing your messages on page sets in this way.

Size of page sets

You should allow enough space in your page sets for the expected peak message capacity. You should also allow for any unexpected peak capacity, such as when a build up of messages develops because a queue server program is not running. This can be done by allocating the page set with secondary extents or, alternatively,

by enabling dynamic page set expansion. For more information, see “Enabling dynamic page set expansion” on page 110.

When planning page set sizes, consider all messages that might be generated in addition to your application’s message data. For example, trigger messages, event messages and any report messages that your application has requested.

The size of the page set determines the time taken to recover a page set when restoring from a backup, because a large page set takes longer to restore.

Note: Recovery of a page set also depends on the time the queue manager takes to process the log records written since the backup was taken; this is determined by the backup frequency. This is discussed in “Recovery procedures” on page 126.

Note: Page sets larger than 4GB require the use of SMS extended addressability.

Calculating the size of your page sets

For queue manager object definitions (for example, queues and processes), it is simple to calculate the storage requirement because these objects are of fixed size and are permanent. For messages however, the calculation is more complex for the following reasons:

- Messages vary in size.
- Messages are transitory.
- Space occupied by messages that have been retrieved is reclaimed periodically by an asynchronous process.

Large page sets of greater than 4GB that provide extra capacity for messages if the network stops, can be created if required. It is not possible to modify the existing page sets. Instead, new page sets with extended addressability and extended format attributes, must be created. The new page sets must be the same physical size as the old ones, and the old page sets must then be copied to the new ones. If backward migration is required, page set zero must not be changed. If page sets less than 4GB are adequate, no action is needed.

Page set zero

For page set zero, the storage required is:

```
   (maximum number of local queue definitions x 1010)
      (excluding shared queues)
+  (maximum number of model queue definitions x 746)
+  (maximum number of alias queue definitions x 338)
+  (maximum number of remote queue definitions x 434)
+  (maximum number of permanent dynamic queue definitions x 1010)
+  (maximum number of process definitions x 674)
+  (maximum number of namelist definitions x 12320)
+  (maximum number of message channel definitions x 2026)
+  (maximum number of client-connection channel definitions x 5170)
+  (maximum number of server-connection channel definitions x 2026)
+  (maximum number of storage class definitions x 266)
+  (maximum number of authentication information definitions x 1010)
+  (maximum number of administrative topic definitions x 15000)
+  (total length of topic strings defined in administrative topic definitions)
```

Divide this value by 4096 to determine the number of records to specify in the cluster for the page set data set.

You do not need to allow for objects that are stored in the shared repository, but you do have to allow for objects that are stored or copied to page set zero (objects with a disposition of GROUP or QMGR).

The total number of objects that you can create is limited by the capacity of page set zero. The number of local queues that you can define is limited to 524 287.

Page sets 01 to 99

For page sets 01 to 99, the storage required for each page set is determined by the number and size of the messages stored on that page set. (Messages on shared queues are not stored on page sets.)

Divide this value by 4096 to determine the number of records to specify in the cluster for the page set data set.

Calculating the storage requirement for messages:

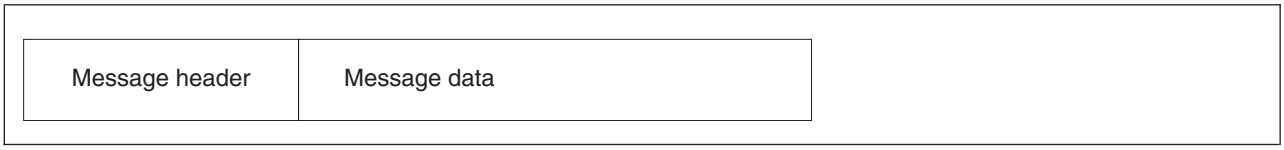
This section describes how messages are stored on pages. Understanding this will help you calculate how much page set storage you need to define for your messages. To calculate the approximate space required for all messages on a page set you must consider maximum queue depth of all the queues that map to the page set and the average size of messages on those queues.

You must allow for the possibility that message "gets" might be delayed for reasons outside the control of WebSphere MQ (for example, because of a problem with your communications protocol). In this case, the "put" rate of messages might far exceed the "get" rate. This could lead to a large increase in the number of messages stored in the page sets and a consequent increase in the storage size demanded.

Each page in the page set is 4096 bytes long. Allowing for fixed header information, each page has 4057 bytes of space available for storing messages.

When calculating the space required for each message, the first thing you need to consider is whether the message fits on one page (a *short message*) or whether it needs to be split over two or more pages (a *long message*). When messages are split in this way, you need to allow for additional control information in your space calculations.

For the purposes of space calculation, a message can be represented like this:



The message header section contains the message descriptor (352 bytes) and other control information, the size of which varies depending on the size of the message. The message data section contains all the actual message data, and any other headers (for example, the transmission header or the IMS bridge header).

A minimum of two pages are required for page set control information. This is typically less than 1% of the total space required for messages.

Short messages:

A short message is defined as a message that fits on one page.

For a short message the control information is 20 bytes long. When this is added to the length of the message header, the usable space remaining on the page is 3685 bytes. If the size of the message data is 3685 bytes or less, WebSphere MQ stores the messages in the next available space on the page, or if there is not enough space available, on the next page, as shown in Figure 19:

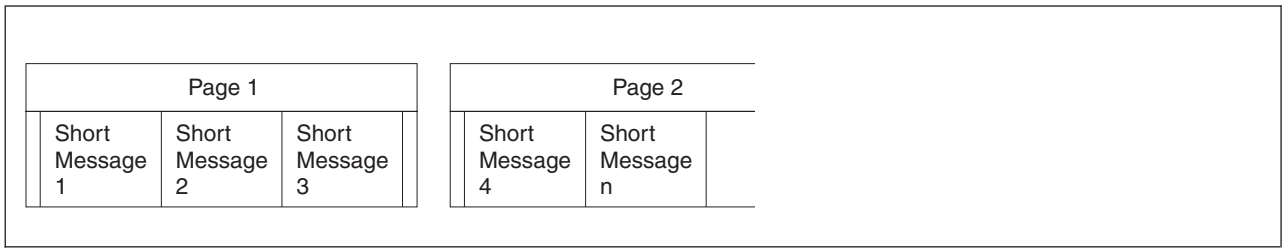


Figure 19. How WebSphere MQ stores short messages on page sets

If there is sufficient space remaining on the page, the next message is also stored on this page, if not, the remaining space on the page is left unused.

Long messages:

If the size of the message data is greater than 3685 bytes, but not greater than 4 MB, the message is classed as a long message. When presented with a long message, WebSphere MQ stores the message on a series of pages, and stores control information that points to these pages in the same way that it would store a short message. This is shown in Figure 20:

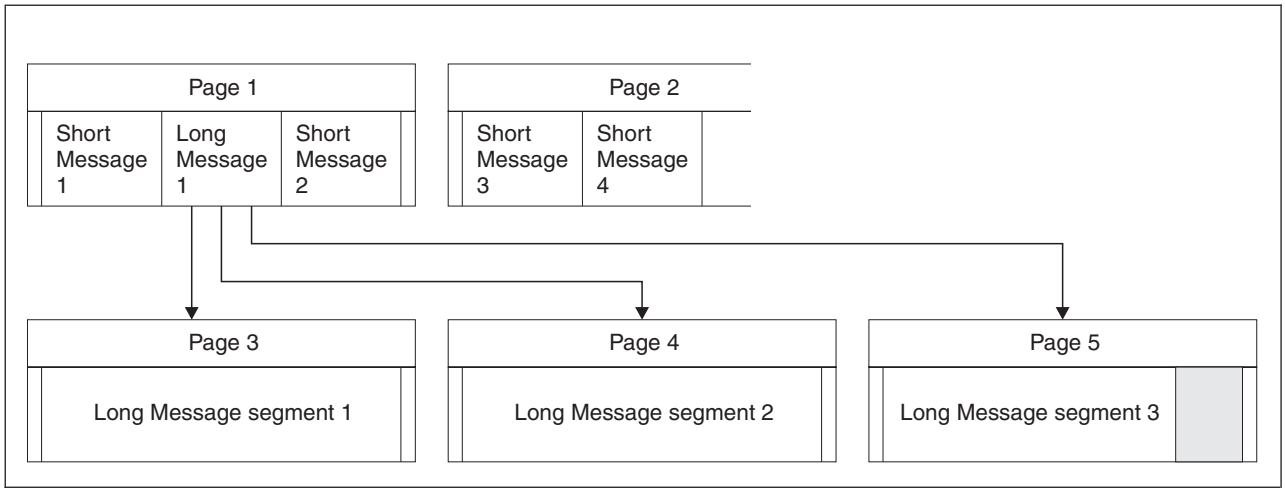


Figure 20. How WebSphere MQ stores long messages on page sets

Each segment of the long message is preceded by 8 bytes of control information, and the first segment also includes the message header portion of 352 bytes. This means that the first page contains 3697 bytes of the message data. The remaining

message data is placed on subsequent pages, in 4049-byte segments. If this does not fill an exact number of pages, the remaining space in the last page is left unused.

The number of pages (n) used for a long message is calculated as follows:

```
     message data length + 352
n = -------------------------
                4049

rounded up to the nearest page
```

In addition to this, you need to allow space for the control information that points to the pages. The length of this (c) depends on the length of the message, and is calculated as follows:

```
c = 20 + (3n) bytes

(where n is the number of pages calculated above)
```

This means that the total page set space required for a long message is:

```
(n * 4096) + c bytes
```

Very long messages:

Very long messages are messages with a size greater than 4 MB. These are stored so that each 4 MB uses 1037 pages. Any remainder is stored in the same way as a long message, as described above.

Enabling dynamic page set expansion

Page sets can be extended dynamically while the queue manager is running. A page set can have 119 extents, and can be spread over multiple disk volumes.

Each time a page set expands, a new data set extent is used. The queue manager continues to expand a page set when required, until the maximum number of extents has been reached, or until no more storage is available for allocation on eligible volumes.

Once page set expansion fails for one of the reasons above, the queue manager marks the page set for no further expansion attempts. This marking can be reset by altering the page set to EXPAND(SYSTEM).

Page set expansion takes place asynchronously to all other page set activity, when 90% of the existing space in the page set is allocated.

The page set expansion process formats the newly allocated extent and makes it available for use by the queue manager. However, none of the space is available for use, until the entire extent has been formatted. This means that expansion by a large extent is likely to take some time, and putting applications might 'block' if they fill the remaining 10% of the page set before the expansion has completed.

Sample thlqual.SCSQPROC(CSQ4PAGE) shows how to define the secondary extents.

To control the size of new extents, you use one of the following options of the EXPAND keyword of the DEFINE PSID and ALTER PSID commands:

- USER
- SYSTEM
- NONE

USER

Uses the secondary extent size specified when the page set was allocated. If a value was not specified, or if a value of zero was specified, dynamic page set expansion cannot occur.

Page set expansion occurs when the space in the page is 90% used, and is performed asynchronously with other page set activity.

This may lead to expansion by more than a single extent at a time.

Consider the following example: you allocate a page set with a primary extent of 100000 pages and a secondary extent of 5000 pages. If you use a maximum 9999 pages, the page set expands when 90000 pages have been used. A second expansion then occurs when page number 94500 has been used (this is because 90% of 105000 is 94500). There have therefore been two expansions and the original allocation is still not full.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set. Only one extent is required to reach this size.

SYSTEM

Ignores the secondary extent size that was specified when the page set was defined. Instead, the queue manager sets a value that is approximately 10% of the current page set size. The value is rounded up to the nearest cylinder of DASD.

If a value was not specified, or if a value of zero was specified, dynamic page set expansion can still occur. The queue manager sets a value that is approximately 10% of the current page set size. The new value is rounded up depending on the characteristics of the DASD.

Page set expansion occurs when the space in the page set is approximately 90% used, and is performed asynchronously with other page set activity.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set.

NONE

No further page set expansion is to take place.

At restart, if a previously used page set has been replaced with a data set that is smaller, it is expanded until it reaches the size of the previously used data set. Only one extent is required to reach this size.

See WebSphere MQ Script (MQSC) Command Reference for more details about the syntax of the DEFINE PSID and ALTER PSID commands.

Defining your buffer pools

You can use up to 16 buffer pools. However, you are recommended to use just the four buffer pools described in Table 19, except in the following circumstances:

- a particular queue is known to require isolation, perhaps because it exhibits significantly different behavior at various times.
  - such a queue might either require the best performance possible under the varying circumstances, or need to be isolated so that they do not adversely impact the other queues in a buffer pool.
  - each such queue can be isolated into its own buffer pool. However, buffer pool 3 (as described below) might be the appropriate place.
- you want to isolate different sets of queues from each other for class of service reasons.
  - each set of queues might then require any or all of the three types of buffer pools 1, 2 and 3, as described in Table 19.

The following table shows suggested values for buffer pool definitions that affect the performance of queue manager operation, recovery, and restart. Two sets of values are given; one set is suitable for a test system, the other for a production system or a system that will become a production system eventually.

Table 19. Suggested definitions for buffer pool settings

Definition setting	Test system	Production system
BUFFPOOL 0	1 050 buffers	50 000 buffers
BUFFPOOL 1	1 050 buffers	20 000 buffers
BUFFPOOL 2	1 050 buffers	50 000 buffers
BUFFPOOL 3	1 050 buffers	20 000 buffers

Reserve buffer pool zero for object definitions (in page set zero) and performance critical, system related message queues, such as the SYSTEM.CHANNEL.SYNCQ queue and the SYSTEM.CLUSTER.* queues. You can use the remaining three buffer pools for user messages, for example:

- You could use buffer pool 1 for important long-lived messages.

  Long-lived messages are those that remain in the system for longer than two checkpoints, at which time they are written out to the page set. If you have a large number of long-lived messages, this buffer pool should be relatively small, so that page set I/O is evenly distributed (older messages are written out to DASD each time the buffer pool becomes 85% full).

  If the buffer pool is too large, page set I/O is delayed until checkpoint processing. This might affect response times throughout the system.

  If you expect a small number of long-lived messages only, define this buffer pool so that it is sufficiently large to hold all these messages.
- You could use buffer pool 2 for performance-critical, short-lived messages.

  There is normally a high degree of buffer reuse, using a small number of buffers; however, you are recommended to make this buffer pool large to allow for unexpected message accumulation, for example, when a server application fails.
- You could use buffer pool 3 for all other (usually performance non-critical) messages. Queues such as the dead-letter queue, SYSTEM.COMMAND.* queues and SYSTEM.ADMIN.* queues can also be mapped to buffer pool 3.

Where virtual storage constraints exist and buffer pools need to be smaller, buffer pool 3 is the first candidate for size reduction.

Initially, define all buffer pools as shown in the table. You can monitor the usage of buffer pools by analyzing buffer pool performance statistics. In particular, you should ensure that the buffer pools are large enough so that the values of QPSTSOS, QPSTSTLA and QPSTNBUF remain at zero. (These performance statistics are described in the WebSphere MQ for z/OS System Setup Guide.)

Tune buffer pool zero and the buffer pool for short-lived messages (buffer pool 2) so that the 15% free threshold is never exceeded (that is, QPSTCBSL divided by QPSTNBUF is always greater than 15%). If more than 15% of buffers remain free, I/O to the page sets using these buffer pools can be largely avoided during normal operation, although messages older than two checkpoints are written to page sets.

Note: The optimum value for these parameters is dependent on the characteristics of the individual system. The values given are intended only as a guideline and might not be appropriate for your system.

MQSeries SupportPac *Capacity planning and tuning for MQSeries for z/OS* gives more information about tuning buffer pools.

Buffer pools can be dynamically re-sized with the ALTER BUFFPOOL command. For more information, see the *Script (MQSC) Command Reference*.

Planning your Coupling Facility and DB2 environment

This chapter discusses the following topics:

Defining Coupling Facility resources

If you intend to use shared queues, you must define the Coupling Facility structures that WebSphere MQ will use in your CFRM policy. To do this you must first update your CFRM policy with information about the structures, and then activate the policy.

Your installation probably has an existing CFRM policy that describes the Coupling Facilities available. The IXCMIAPU z/OS utility is used to modify the contents of the policy based on textual statements you provide. The utility is described in the *MVS Setting up a Sysplex* manual. You must add statements to the policy that define the names of the new structures, the Coupling Facilities that they are defined in, and what size the structures are.

Planning your structures

A queue-sharing group requires a minimum of two structures to be defined. The first structure, known as the administrative structure, is used to coordinate WebSphere MQ internal activity across the queue-sharing group. No user data is held in this structure. It has a fixed name of *qsg-name*CSQ_ADMIN (where *qsg-name* is the name of your queue-sharing group). Subsequent structures are used to hold the messages on WebSphere MQ shared queues. Each structure can hold up to 512 shared queues.

Using multiple structures:

A queue-sharing group can connect to up to 64 Coupling Facility structures. One of these structures must be the administration structure, but you can use up to 63 structures for WebSphere MQ data. You might choose to use multiple structures for any of the following reasons:

- You have some queues that are likely to hold a very large number of messages and so require all the resources of an entire Coupling Facility.
- You have a requirement for a very large number of shared queues, so they must be split across multiple structures because each structure can contain only 512 queues.
- RMF™ reports on the usage characteristic of a structure suggest that you should distribute the queues it contains across a number of Coupling Facilities.
- You want some queue data to held in a physically different Coupling Facility from other queue data for data isolation reasons.
- Recovery of persistent shared messages is performed using structure level attributes and commands, for example BACKUP CFSTRUCT. To simplify backup and recovery, you could assign queues that hold nonpersistent messages to different structures from those that hold persistent messages.

When choosing which Coupling Facilities to allocate the structures in, consider the following points:

- Your data isolation requirements.
- The volatility of the Coupling Facility (that is, its ability to preserve data through a power outage).
- Failure independence between the accessing systems and the Coupling Facility, or between Coupling Facilities.
- The level of Coupling Facility Control Code (CFCC) installed on the Coupling Facility (WebSphere MQ requires Level 9 or higher).

Planning the size of your structures

The administrative structure (*qsg-name*CSQ_ADMIN) must be large enough to contain 1000 list entries for each queue manager in the queue-sharing group. When a queue manager starts, the structure is checked to see if it is large enough for the number of queue managers currently *defined* to the queue-sharing group. Queue managers are considered as being defined to the queue-sharing group if they have been added by the CSQ5PQSG utility. You can check which queue managers are defined to the group with the MQSC DISPLAY GROUP command.

Table 20 shows the minimum required size for the administrative structure for various numbers of queue managers defined in the queue-sharing group. These sizes were established for a CFCC level 14 Coupling Facility structure; for higher levels of CFCC, they probably need to be larger.

Table 20. Minimum administrative structure sizes

Number of queue managers defined in queue-sharing group	Required storage
1	6144 KB
2	6912 KB
3	7976 KB
4	8704 KB

Table 20. Minimum administrative structure sizes (continued)

Number of queue managers defined in queue-sharing group	Required storage
5	9728 KB
6	10496 KB
7	11520 KB
8	12288 KB
9	13056 KB
10	14080 KB
11	14848 KB
12	15616 KB
13	16640 KB
14	17408 KB
15	18176 KB
16	19200 KB
17	19968 KB
18	20736 KB
19	21760 KB
20	22528 KB
21	23296 KB
22	24320 KB
23	25088 KB
24	25856 KB
25	27136 KB
26	27904 KB
27	28672 KB
28	29696 KB
29	30464 KB
30	31232 KB
31	32256 KB

When you add a queue manager to an existing queue-sharing group, the storage requirement might have increased beyond the size recommended in Table 20 on page 114. If this is the case, use the following procedure to estimate the required storage for the CSQ_ADMIN structure: Issue MQSC command `+cpf DISPLAY CFSTATUS(*)`, where +cpf is for an existing member of the queue-sharing group, and extract the ENTSMAX information for the CSQ_ADMIN structure. If this number is less than 1000 times the total number of queue managers you want to define in the queue-sharing group (as reported by the DISPLAY GROUP command), increase the structure size.

The size of the structures required to hold WebSphere MQ messages depends on the likely number and size of the messages to be held on a structure concurrently, together with an estimate of the likely number of concurrent units of work.

The graph in Figure 21 shows how large you should make your CF structures to hold the messages on your shared queues. To calculate the allocation size you need to know

- The average size of messages on your queues
- The total number of messages likely to be stored in the structure

Find the number of messages along the horizontal axis. (Ticks are at multiples of 2, 5, and 8.) Select the curve that corresponds to your message size and determine the required value from the vertical axis. For example, for 200 000 messages of length 1 KB gives a value between 256 and 512MB.

Table 21 provides the same information in tabular form.

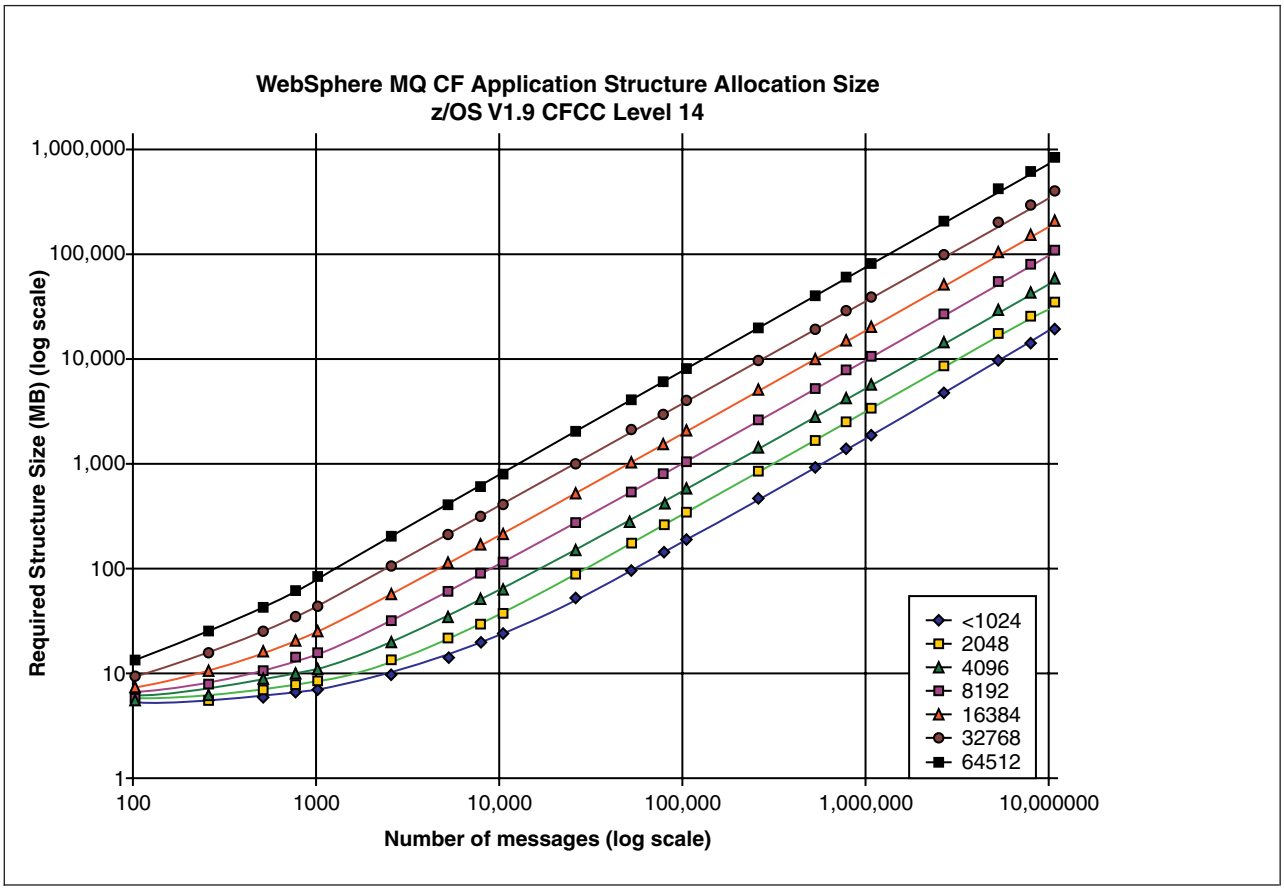


Figure 21. Calculating the size of a Coupling Facility structure

Use this table to help calculate how large to make your Coupling Facility structures:

Table 21. Calculating the size of a Coupling Facility structure

Number of messages	1 KB	2 KB	4 KB	8 KB	16 KB	32 KB	63 KB
100	6	6	7	7	8	10	14
1000	8	9	12	17	27	48	88
10000	25	38	64	115	218	423	821
100000	199	327	584	1097	2124	4177	8156

Your CFRM policy should include the following statements:

INITSIZE is the size in KB that XES allocates to the structure when the first connector connects to it. SIZE is the maximum size that the structure can attain. FULLTHRESHOLD sets the percentage value of the threshold at which XES issues message IXC585E to indicate that the structure is getting full.

For example, with the figures determined above, you might include the following statements:

```
STRUCTURE NAME(structure-name)
INITSIZE(value from graph in KB, that is, multiplied by 1024)
SIZE(something larger)
FULLTHRESHOLD(85)
```

```
STRUCTURE NAME(QSG1APPLICATION1)
INITSIZE(272144) /* 256 MB */
SIZE(524288)  /* 512 MB */
FULLTHRESHOLD(85)
```

If the structure use reaches the threshold where warning messages are issued, intervention is required. You might use WebSphere MQ to inhibit **MQPUT** operations to some of the queues in the structure to prevent applications from writing more messages, start more applications to get messages from the queues, or quiesce some of the applications that are putting messages to the queue.

Alternatively, you can use XES facilities to alter the structure size in place. The following z/OS command:

```
SETXCF START,ALTER,STRNAME=structure-name,SIZE=newsize
```

alters the size of the structure to *newsize*, where *newsize* is a value that is less than the value of SIZE specified on the CFRM policy for the structure, but greater than the current Coupling Facility size.

You can monitor the use of a Coupling Facility structure with the MQSC DISPLAY GROUP command.

If no action is taken and a queue structure fills up, an MQRC_STORAGE_MEDIUM_FULL return code is returned to the application. If the administration structure becomes full, the exact symptoms depend on which processes experience the error, but they might include the following problems:

- No responses to commands.
- Queue manager failure as a result of problems during commit processing.

Mapping shared queues to structures

The CFSTRUCT attribute of the queue definition is used to map the queue to a structure.

WebSphere MQ adds the name of the queue-sharing group to the beginning of the CFSTRUCT attribute. For a structure defined in the CFRM policy with name *qsg-name*SHAREDQ01, the definition of a queue that uses this structure would be:

```
DEFINE QLOCAL(myqueue) QSGDISP(SHARED) CFSTRUCT(SHAREDQ01)
```

Planning your DB2 environment

If you are using queue-sharing groups, WebSphere MQ needs to attach to a DB2 subsystem that is a member of a data-sharing group. WebSphere MQ needs to know the name of the data-sharing group that it is to connect to, and the name of a DB2 subsystem (or DB2 group) to connect to, to reach this data-sharing group. These names are specified in the QSGDATA parameter of the CSQ6SYSP system parameter macro (described in the WebSphere MQ for z/OS System Setup Guide).

WebSphere MQ uses the RRS Attach facility of DB2. This means that you can specify the name of a DB2 group that you want to connect to. The advantage of connecting to a DB2 group attach name (rather than a specific DB2 subsystem), is that WebSphere MQ can connect (or reconnect) to any available DB2 subsystem on the z/OS image that is a member of that group. There must be a DB2 subsystem that is a member of the data-sharing group active on each z/OS image where you are going to run a queue-sharing WebSphere MQ subsystem, and RRS must be active.

DB2 storage

You need to set up a set of DB2 tables that are used to hold WebSphere MQ data. These are automatically defined for you by WebSphere MQ when you set up your DB2 environment.

(This is described in the WebSphere MQ for z/OS System Setup Guide).

For most installations, the amount of DB2 storage required is about 20 or 30 cylinders on a 3390 device. However, if you want to calculate your storage requirement, the following table gives some information to help you determine how much storage DB2 requires for the WebSphere MQ data. The table describes the length of each DB2 row, and when each row is added to or deleted from the relevant DB2 table. Use this information together with the information about calculating the space requirements for the DB2 tables and their indexes in the *DB2 for z/OS Installation Guide*.

Table 22. Planning your DB2 storage requirements

DB2 table name	Length of row	A row is added when:	A row is deleted when:
CSQ.ADMIN_B_QSG	252 bytes	A queue-sharing group is added to the table with the ADD QSG function of the CSQ5PQSG utility.	A queue-sharing group is removed from the table with the REMOVE QSG function of the CSQ5PQSG utility. (All rows relating to this queue-sharing group are deleted automatically from all the other DB2 tables when the queue-sharing group record is deleted.)
CSQ.ADMIN_B_QMGR	Up to 3828 bytes	A queue manager is added to the table with the ADD QMGR function of the CSQ5PQSG utility.	A queue manager is removed from the table with the REMOVE QMGR function of the CSQ5PQSG utility.
CSQ.ADMIN_B_STRUCTURE	329 bytes	The first local queue definition, specifying the QSGDISP(SHARED) attribute, that names a previously unknown structure within the queue-sharing group is defined.	The last local queue definition, specifying the QSGDISP(SHARED) attribute, that names a structure within the queue-sharing group is deleted.

Table 22. Planning your DB2 storage requirements (continued)

DB2 table name	Length of row	A row is added when:	A row is deleted when:
CSQ.ADMIN_B_SCST	342 bytes	A shared channel is started.	A shared channel becomes inactive.
CSQ.ADMIN_B_SSKT	254 bytes	A shared channel that has the NPMSPEED(NORMAL) attribute is started.	A shared channel that has the NPMSPEED(NORMAL) attribute becomes inactive.
CSQ.ADMIN_B_STRBACKUP	507 bytes	A new row is added to the CSQ.ADMIN_B_STRUCTURE table. Each entry is a dummy entry until the BACKUP CFSTRUCT command is run, which overwrites the dummy entries.	A row is deleted from the CSQ.ADMIN_B_STRUCTURE table.
CSQ.OBJ_B_AUTHINFO	3400 bytes	An authentication information object with QSGDISP(GROUP) is defined.	An authentication information object with QSGDISP(GROUP) is deleted.
CSQ.OBJ_B_QUEUE	Up to 3707 bytes	• A queue with the QSGDISP(GROUP) attribute is defined. • A queue with the QSGDISP(SHARED) attribute is defined. • A model queue with the DEFTYPE(SHAREDYN) attribute is opened.	• A queue with the QSGDISP(GROUP) attribute is deleted. • A queue with the QSGDISP(SHARED) attribute is deleted. • A dynamic queue with the DEFTYPE(SHAREDYN) attribute is closed with the DELETE option.
CSQ.OBJ_B_NAMELIST	Up to 15127 bytes	A namelist with the QSGDISP(GROUP) attribute is defined.	A namelist with the QSGDISP(GROUP) attribute is deleted.
CSQ.OBJ_B_CHANNEL	Up to 14127 bytes	A channel with the QSGDISP(GROUP) attribute is defined.	A channel with the QSGDISP(GROUP) attribute is deleted.
CSQ.OBJ_B_STGCLASS	Up to 2865 bytes	A storage class with the QSGDISP(GROUP) attribute is defined.	A storage class with the QSGDISP(GROUP) attribute class is deleted.
CSQ.OBJ_B_PROCESS	Up to 3347 bytes	A process with the QSGDISP(GROUP) attribute is defined.	A process with the QSGDISP(GROUP) attribute is deleted.
CSQ.EXTEND_B_QMGR	Less than 430 bytes	A queue manager is added to the table with the ADD QMGR function of the CSQ5PQSG utility.	A queue manager is removed from the table with the REMOVE QMGR function of the CSQ5PQSG utility.
CSQ.ADMIN_B_MESSAGE	87 bytes	For large message PUT (1 per BLOB).	For large message GET (1 per BLOB).
CSQ.ADMIN_MSGS_BAUX1 CSQ.ADMIN_MSGS_BAUX2 CSQ.ADMIN_MSGS_BAUX3 CSQ.ADMIN_MSGS_BAUX4		These 4 tables contain message payload for large messages added into one of these 4 tables for each BLOB of the message. BLOBS are up to 511K in length, so if the message size is > 711K, there will be multiple BLOBs for this message.	

The use of large numbers of shared queue messages of size greater than 63KB can have significant performance implications on your WebSphere MQ system. For more information, see SupportPac MP16, Capacity Planning and Tuning for WebSphere MQ for z/OS, at: www.ibm.com/software/integration/support/supportpacs/

Planning your logging environment

The WebSphere MQ logging environment is established using the system parameter macros to specify options, such as whether to have single or dual active logs, what media to use for the archive log volumes, and how many log buffers to have. These macros are described in the WebSphere MQ for z/OS System Setup Guide.

This chapter discusses the following topics:

- "Planning your logs"
- "Logs and archive storage" on page 123
- "Planning your archive storage" on page 124

Planning your logs

You are recommended to have at least three log data sets, and to use dual logging and log archiving. The following table shows suggested values for log and bootstrap data set definitions that affect the performance of queue manager operation, recovery, and restart. Two sets of values are given; one set is suitable for a test system, the other for a production system or a system that will become a production system.

Table 23. Suggested definitions for log and bootstrap data sets

Definition setting	Test system	Production system
Log data set size[1]	10 000 records (40 MB approx.) Access Method Services rounds to nearest cylinder and allocates 10 080 records.	180 000 records (700 MB or 1 000 cylinders of 3390)
Number of active logs[1]	3	6
BSDS size	60 records (240 KB approx.)	120 records (480 KB approx.) A primary extent of this size should be sufficient for the maximum number of archive logs that can be recorded in the BSDS (1 000).

Note:

1. The optimum value for this definition is dependent on the characteristics of the individual system. The values given are intended as a guideline only and might not be appropriate for your system.

Note: If you are using queue-sharing groups, ensure that you define the bootstrap and log data sets with SHAREOPTIONS(2 3).

Log data set definitions

Before setting up the log data sets, review the following section to decide on the most appropriate configuration for your system.

Should your installation use single or dual logging?:

Either single or dual logging might be appropriate, depending on your situation.

With single logging data is written to one set of log data sets. With dual logging data is written to two sets of log data sets, so in the event of a problem with one log dataset, such as the data set being accidentally deleted, the equivalent dataset in the other set of logs can be used to recover the data.

If you are using dual logging, you should also use dual BSDSs and dual archiving to ensure adequate provision for data recovery.

Dual active logging adds a small performance overhead.

Attention: Always use dual logging and dual BSDSs rather than dual writing to DASD (mirroring). If a mirrored data set is accidentally deleted, both copies are lost.

If you use persistent messages, single logging can increase maximum capacity by 10-30% and can also improve response times.

Single logging uses 2 - 31 active log data sets, whereas dual logging uses 4 - 62 to provide the same number of active logs. Thus single logging reduces the amount of data logged, which might be important if your installation is I/O constrained.

How many active log data sets do you need?:

Define sufficient active logs to ensure that your system is not impacted in the event of an archive being delayed.

In practice, you should have at least three active log data sets but it is preferable to define more. For example, if the time taken to fill a log is likely to approach the time taken to archive a log during peak load, define more logs. You are also recommended to define more logs to offset possible delays in log archiving. If you use archive logs on tape, allow for the time required to mount the tape.

Consider having enough active log space to keep a day's worth of data, in case the system is unable to archive because of lack of DASD or because it cannot write to tape.

It is possible to dynamically define new active log data sets as a way of minimizing the effect of archive delays or problems. New data sets can be brought online rapidly, in order to avoid queue manager 'stall' due to lack of space in the active log.

How large should the active logs be?:

Make your logs large enough so that it takes at least 30 minutes to fill a single log during peak message loads.

If you are archiving to tape, make the logs large enough to fill one tape cartridge, or a number of tape cartridges. (For example, a log size of 1000 cylinders on 3390 DASD fits onto a 3490E non-compacted tape with space to spare.)

Note: When archiving to tape, a copy of the BSDS is also written to the tape. When archiving to DASD, a separate data set is created for the BSDS copy.

If the logs are small (for example, 10 cylinders) it is likely that they will fill up frequently, which might result in performance degradation. In addition, you might find that the large number of archive logs required is difficult to manage.

If the logs are very large, and you are archiving to DASD, you need a corresponding amount of space reserved on DASD for SMS retrieval of migrated archive logs, which might cause space management problems. In addition, the time taken to restart might increase because one or more of the logs has to be read sequentially at startup.

Active log placement:

If you are using RAID (Redundant Array of Independent Disks) devices then data set placement is not critical, as you have little control over where data is written in the subsystem.

If you are not using RAID, for example you are using 3390 DASD, then plan carefully where you place your data sets to give optimum performance. Consider the following points:

- Place active log data sets on low usage devices.
- If possible, allocate each active log on a separate DASD volume. As a minimum, no two adjacent logs should be on the same volume.

  When an active log fills, the next log in the ring is used and the previous log data set is copied to the archive data set. If these two active data sets are on the same volume, contention will result, because one data set is read while the other is written to. For example, if you have three active logs and use dual logging, you need six DASD volumes because each log is adjacent to both of the two other logs. Alternatively, if you have four active logs and you want to conserve DASD space, by allocating logs 1 and 3 on one volume and logs 2 and 4 on another, you need four DASD volumes only.
- Ensure that primary and secondary logs are on separate physical units. If you use 3390 DASD, be aware that each head disk assembly contains two logical volumes. The physical layout of other DASD subsystems such as RAMAC® arrays should also be taken into account.
- Allocate archive log data sets on different devices to the active log data sets to prevent contention if an active log and archive log are being written to at the same time.
- If you use dual logging, ensure that each set of active and archive logs is kept apart. For example, allocate them on separate DASD subsystems, or on different devices.

  This reduces the risk of them both being lost if one of the volumes is corrupted or destroyed. If both copies of the log are lost, the probability of data loss is high.

You can improve log write performance by pre-formatting the logs using the CSQJUFMT utility.

Logs and archive storage

Active log data sets record significant events and data changes. They are periodically off-loaded to the archive log. Consequently, the space requirements for your active log data sets depend on the volume of messages that your WebSphere MQ handles and how often the active logs are off-loaded to your archive data sets. WebSphere MQ provides optional support for dual logging; if you use this your log storage requirement doubles.

If you decide to place the archive data sets on direct access storage devices (DASD), you need to reserve enough space on the devices. You should also reserve space for the bootstrap data sets (BSDS). A typical size for each BSDS might be 500 KB. These are all separate data sets and you should allocate space for them on different volumes and strings if possible to minimize DASD contention and problems caused by any defects on the physical devices.

Because each change to the system is logged, you can estimate the size of storage required from the size and expected throughput of persistent messages (nonpersistent messages are not logged). You must add to this a small overhead for the header information in the data sets.

Additionally, CF structure backups are written to the active log of the queue manager where the BACKUP CFSTRUCT command is issued.

To arrive at the size of the log extents, you can develop an algorithm that depends on various factors including the message rate and size of persistent messages and how frequently you want to switch the log, and CF structure backup characteristics.

Figure 22 shows an approximate calculation for the number of records to specify in the cluster for the log data set.

```
Number of records = ((a * log switch interval) + S) / 4096

where a = (Number of puts/sec*(average persistent message size+500))
         + (Number of gets/sec*110)
         + (Number of units of recovery started*120)
         + (Number of syncpoints a second*240)
 and
       log switch interval = time period between successive log
                             switches required in seconds
 and
       S = the number of CF structure backups in the log switch
       interval multiplied by the average structure size
```

Figure 22. Calculating the number of records to specify in the cluster for the log data set

Each log data set should have the same number of records specified and should not have secondary extents. Other than for a very small number of records, AMS rounds up the number of records so that a whole number of cylinders is allocated. The number of records actually allocated is:

```
c = (INT (number of log records / b) + 1) * b

Where b is the number of 4096-byte blocks in each cylinder (180 for a
3390 device) and INT means round down to an integer
```

Planning your archive storage

This section describes the different ways of maintaining your archive log data sets.

You can place archive log data sets on standard-label tapes, or DASD, and you can manage them by data facility hierarchical storage manager (DFHSM). Each z/OS logical record in an archive log data set is a VSAM control interval from the active log data set. The block size is a multiple of 4 KB.

Archive log data sets are dynamically allocated, with names chosen by WebSphere MQ. The data set name prefix, block size, unit name, and DASD sizes needed for such allocations are specified in the system parameter module. You can also choose, at installation time, to have WebSphere MQ add a date and time to the archive log data set name.

It is not possible to choose specific volumes for new archive logs. If allocation errors occur, off-loading is postponed until the next time off-loading is triggered.

If you specify dual archive logs at installation time, each log control interval retrieved from the active log is written to two archive log data sets. The log records that are contained in the pair of archive log data sets are identical, but the end-of-volume points are not synchronized for multivolume data sets.

Should your archive logs reside on tape or DASD?

When deciding whether to use tape or DASD for your archive logs, there are a number of factors that you should consider:

- Review your operating procedures before making decisions about tape or disk. For example, if you choose to archive to tape, operators must be available to mount the appropriate tapes when they are required.
- During recovery, archive logs on tape are available as soon as the tape is mounted. If DASD archives have been used, and the data sets migrated to tape using hierarchical storage manager (HSM), there is a delay while HSM recalls each data set to disk. You can recall the data sets before the archive log is used. However, it is not always possible to predict the correct order in which they are required.
- When using archive logs on DASD, if many logs are required (which might be the case when recovering a page set after restoring from a backup) you might require a significant quantity of DASD to hold all the archive logs.
- In a low-usage system or test system, it might be more convenient to have archive logs on DASD to eliminate the need for tape mounts.
- Both issuing a RECOVER CFSTRUCT command and backing out a persistent unit of work result in the log being read backwards. Tape drives with hardware compression perform very badly on operations that read backwards. You should plan sufficient log data on DASD to avoid reading backwards from tape.

Archiving to DASD offers faster recoverability but is more expensive than archiving to tape. If you use dual logging, you can specify that the primary copy of the archive log go to DASD and the secondary copy go to tape. This increases recovery speed without using as much DASD, and you can use the tape as a backup.

Archiving to tape:

If you choose to archive to a tape device, WebSphere MQ can extend to a maximum of twenty volumes.

If you choose to off-load to tape, you should consider adjusting the size of your active log data sets so that each nearly fills a tape volume. This minimizes tape handling and volume mounts, and maximizes the use of tape resources. However, such an adjustment is not essential.

If you are considering changing the size of the active log data set so that the set fits on one tape volume, you must bear in mind that a copy of the BSDS is placed on the same tape volume as the copy of the active log data set. Adjust the size of the active log data set downward to offset the space required for the BSDS on the tape volume.

If you use dual archive logs on tape, it is typical for one copy to be held locally, and the other copy to be held off-site for use in disaster recovery.

Archiving to DASD volumes:

WebSphere MQ requires that you catalog all archive log data sets allocated on non-tape devices (DASD). If you choose to archive to DASD, the CATALOG parameter of the CSQ6ARVP macro must be YES. (This macro is described in the WebSphere MQ for z/OS System Setup Guide.) If this parameter is NO, and you decide to place archive log data sets on DASD, you receive message CSQJ072E each time an archive log data set is allocated, although WebSphere MQ still catalogs the data set.

If the archive log data set is held on DASD, the archive log data sets cannot extend to another volume.

If you choose to use DASD, make sure that the primary space allocation (both quantity and block size) is large enough to contain either the data coming from the active log data set, or that from the corresponding BSDS, whichever is the larger of the two. This minimizes the possibility of unwanted z/OS X'B37' or X'E37' abends during the off-load process. The primary space allocation is set with the PRIQTY (primary quantity) parameter of the CSQ6ARVP macro.

Using SMS with archive log data sets:

If you have MVS/DFP storage management subsystem (DFSMS™) installed, you can write an Automatic Class Selection (ACS) user-exit filter for your archive log data sets, which helps you convert them for the SMS environment. Such a filter, for example, can route your output to a DASD data set, which DFSMS can manage. You must exercise caution if you use an ACS filter in this manner. Because SMS requires DASD data sets to be cataloged, you must make sure the CATALOG DATA field of the CSQ6ARVP macro contains YES. If it does not, message CSQJ072E is returned; however, the data set is still cataloged by WebSphere MQ.

For more information about ACS filters, see the *DFP Storage Administration Reference* manual, and the *SMS Migration Planning Guide*.

Planning for backup and recovery

Developing backup and recovery procedures at your site is vital to avoid costly and time-consuming losses of data. WebSphere MQ provides means for recovering both queues and messages to their current state after a system failure.

This chapter contains the following sections:

Recovery procedures

You should develop the following procedures for WebSphere MQ:

- Creating a point of recovery.
- Backing up page sets.
- Backing up CF structures.
- Recovering page sets.
- Recovering from out-of-space conditions (WebSphere MQ logs and page sets).
- Recovering CF structures.

See the WebSphere MQ for z/OS System Administration Guide for information about these.

You should also be familiar with the procedures used at your site for the following:

- Recovering from a hardware or power failure.
- Recovering from a z/OS component failure.
- Recovering from a site interruption, using off-site recovery.

Tips for backup and recovery

This section introduces some backup and recovery tasks. The queue manager restart process recovers your data to a consistent state by applying log information to the page sets. If your page sets are damaged or unavailable, you can resolve the problem using your backup copies of your page sets (provided that all the logs are available). If your log data sets are damaged or unavailable, it might not be possible to recover completely.

Periodically take backup copies

A *point of recovery* is the term used to describe a set of backup copies of WebSphere MQ page sets and the corresponding log data sets required to recover these page sets. These backup copies provide a potential restart point in the event of page set loss (for example, page set I/O error). If you restart the queue manager using these backup copies, the data in WebSphere MQ will be consistent up to the point that these copies were taken. Providing that all logs are available from this point, WebSphere MQ can be recovered to the point of failure.

The more recent your backup copies, the quicker WebSphere MQ can recover the data in the page sets. The recovery of the page sets are dependent on all the necessary log data sets being available.

In planning for recovery, you need to determine how often to take backup copies and how many complete backup cycles to keep. These values tell you how long you must keep your log data sets and backup copies of page sets for WebSphere MQ recovery.

When deciding how often to take backup copies, consider the time needed to recover a page set. The time needed is determined by the following:

- The amount of log to traverse.
- The time it takes an operator to mount and remove archive tape volumes.
- The time it takes to read the part of the log needed for recovery.
- The time needed to reprocess changed pages.
- The storage medium used for the backup copies.
- The method used to make and restore backup copies.

In general, the more frequently you make backup copies, the less time recovery takes, but the more time is spent making copies.

For each queue manager, you should take backup copies of the following:

- The archive log data sets
- The BSDS copies created at the time of the archive
- The page sets
- Your object definitions
- Your CF structures

To reduce the risk of your backup copies being lost of damaged, you should consider:

- Storing the backup copies on different storage volumes to the original copies.
- Storing the backup copies at a different site to the original copies.
- Making at least two copies of each backup of your page sets and, if you are using single logging or a single BSDS, two copies of your archive logs and BSDS. If you are using dual logging or BSDS, make a single copy of both archive logs or BSDS.

Before moving WebSphere MQ to a production environment you should have tested and documented your backup procedures.

Backing up your object definitions:

You should also create backup copies of your object definitions. To do this, use the MAKEDEF feature of the COMMAND function of the utility program (described in the WebSphere MQ for z/OS System Administration Guide).

You should do this whenever you take backup copies of your queue manager data sets, and keep the most current version.

Backing up your Coupling Facility structures:

If you have set up any queue-sharing groups, even if you are not using them, you must take periodic backups of your CF structures. To do this, use the WebSphere MQ BACKUP CFSTRUCT command (described in the WebSphere MQ Script (MQSC) Command Reference). You can use this command only on CF structures that are defined with the RECOVER(YES) attribute.

It is recommended that you take a backup of all your CF structures about every hour, to minimize the time it takes to restore a CF structure.

You could perform all your CF structure backups on a single queue manager, which has the advantage of limiting the increase in log use to a single queue manager. Alternatively, you could perform backups on all the queue managers in the queue-sharing group, which has the advantage of spreading the workload across the queue-sharing group. Whichever strategy you use, WebSphere MQ can locate the backup and perform a RECOVER CFSTRUCT from any queue manager in the queue-sharing group. The logs of all the queue managers in the queue-sharing group need to be accessed to recover the CF structure.

Do not discard archive logs you might need

WebSphere MQ might need to use archive logs during restart. You must keep sufficient archive logs so the system can be fully restored. WebSphere MQ might use an archive log to recover a page set from a restored backup copy. If you have discarded that archive log, WebSphere MQ cannot restore the page set to its current state. When and how you should discard archive logs is described in the WebSphere MQ for z/OS System Administration Guide.

Do not change the DDname to page set association

WebSphere MQ associates page set number 00 with DDname CSQP0000, page set number 01 with DDname CSQP0001, and so on up to CSQP0099. WebSphere MQ writes recovery log records for a page set based on the DDname that the page set is associated with. For this reason, you must not move page sets that have already been associated with a PSID DDname.

Recovering page sets

A key factor in recovery strategy concerns the period of time for which you can tolerate a queue manager outage. The total outage time might include the time taken to recover a page set from a backup, or to restart the queue manager after an abnormal termination. Factors affecting restart time include how frequently you back up your page sets, and how much data is written to the log between checkpoints.

To minimize the restart time after an abnormal termination, keep units of work short so that, at most, two active logs are used when the system restarts. For example, if you are designing a WebSphere MQ application, avoid placing an

MQGET call that has a long wait interval between the first in-syncpoint MQI call and the commit point because this might result in a unit of work that has a long duration. Another common cause of long units of work is batch intervals of more than 5 minutes for the channel initiator.

You can use the DISPLAY THREAD command to display the RBA of units of work and to help resolve the old ones. For information about the DISPLAY THREAD command, see the WebSphere MQ Script (MQSC) Command Reference manual.

How often should a page set be backed up?

Frequent page set backup is essential if a reasonably short recovery time is required. This applies even when a page set is very small or there is a small amount of activity on queues in that page set.

If you use persistent messages in a page set, the backup frequency should be in hours rather than days. This is also the case for page set zero.

To calculate an approximate backup frequency, start by determining the target total recovery time. This consists of the following:

1. The time taken to react to the problem.
2. The time taken to restore the page set backup copy.

   (For example, you can restore approximately 60 cylinders of 3390 data a minute from and to RAMAC Virtual Array 2 Turbo 82 (RVA2-T82) DASD using Access Method Services REPRO.)

   If you use SnapShot backup/restore, the time taken to perform this task is a few seconds. For information about SnapShot, see the *DFSMSdss™ Storage Administration Guide*.
3. The time the queue manager requires to restart, including the additional time needed to recover the page set.

   This depends most significantly on the amount of log data that must be read from active and archive logs since that page set was last backed up. All such log data must be read, in addition to that directly associated with the damaged page set.

   Note: When using *fuzzy backup* (where a snapshot is taken of the logs and page sets while a unit of work is active), it might be necessary to read up to three additional checkpoints, and this might result in the need to read one or more additional logs.

When deciding on how long to allow for the recovery of the page set, the factors that you need to consider are:

- The rate at which data is written to the active logs during normal processing:
  - Approximately 1.3 KB of extra data is required on the log for a persistent message.
  - Approximately 2.5 KB of data is required on the log for each batch of messages sent on a channel.
  - Approximately 1.4 KB of data is required on the log for each batch of messages received on a channel.
  - Nonpersistent messages require no log data. NPMSPEED(FAST) channels require no log data for batches consisting entirely of nonpersistent messages.

The rate at which data is written to the log depends on how messages arrive in your system, in addition to the message rate. Messages received or sent over a channel result in more data logging than messages generated and retrieved locally.

- The rate at which data can be read from the archive and active logs.

  When reading the logs, the achievable data rate depends on the devices used and the total load on your particular DASD subsystem. (For example, data rates of approximately 2.7 MB a second have been observed using active and archive logs on RVA2-T82 DASD.)

  With most tape units, it is possible to achieve higher data rates for archived logs with a large block size. However, if an archive log is required for recovery, all the data on the active logs must be read also.

Recovering CF structures

At least one queue manager in the queue-sharing group must be active to process a RECOVER CFSTRUCT command. CF structure recovery does not impact queue manager restart time, because recovery is performed by an already active queue manager.

The recovery process consists of two logical steps that are managed by the RECOVER CFSTRUCT command:

1. Locating and restoring the backup.
2. Merging all the logged updates to persistent messages that are held on the CF structure from the logs of all the queue managers in the queue-sharing group that have used the CF structure, and applying the changes to the backup.

The second step is likely to take much longer because a lot of log data might need to be read. You can reduce the time taken if you take frequent backups, or if you recover multiple CF structures at the same time, or both.

The queue manager performing the recovery locates the relevant backups on all the other queue managers' logs using the data in DB2 and the bootstrap data sets. The queue manager replays these backups in the correct time sequence across the queue sharing group, from just before the last backup through to the point of failure.

The time it takes to recover a CF structure depends on the amount of recovery log data that must be replayed, which in turn depends on the frequency of the backups. In the worst case, it takes as long to read a queue manager's log as it did to write it. So if, for example, you have a queue-sharing group containing six queue managers, an hour's worth of log activity could take six hours to replay. In general it takes less time than this, because reading can be done in bulk, and because the different queue manager's logs can be read in parallel. As a starting point, we recommend that you backup your CF structures every hour.

All queue managers can continue working with non-shared queues and queues in other CF structures while there is a failed CF structure. However, if the administration structure has also failed, all the queue managers in the queue-sharing group must be started before you can issue the RECOVER CFSTRUCT.

Backing up CF structures can require considerable log writing capacity, and can therefore impose a large load on the queue manager doing the backup. Choose a

lightly-loaded queue manager for doing backups; for very busy systems, add an additional queue manager to the queue-sharing group and dedicate it exclusively for doing backups.

Achieving specific recovery targets

If you have specific recovery targets to achieve, for example, completion of the queue manager recovery and restart processing in addition to the normal startup time within *xx* seconds, you can use the following calculation to estimate your backup frequency (in hours):

```
Formula (A)
                    Required restart time * System recovery log read rate
                          (in secs)                 (in MB/sec)
Backup frequency  = ------------------------------------------------------
   (in hours)              Application log write rate (in MB/hour)
```

Note: The examples given below are intended to highlight the need to back up your page sets frequently. The calculations assume that the majority of log activity is derived from a large number of persistent messages. However, there are situations where the amount of log activity is not easily calculated. For example, in a queue-sharing group environment, a unit of work in which shared queues are updated in addition to other resources might result in UOW records being written to the WebSphere MQ log. For this reason, the 'Application log write rate' in Formula (A) can be derived accurately only from the observed rate at which the WebSphere MQ logs fill.

For example, consider a system in which WebSphere MQ clients generate a total load of 100 persistent messages a second. In this case, all messages are generated locally.

If each message is of user length 1 KB, the amount of data logged each hour is approximately:

```
100 * (1 + 1.3) KB * 3600 = approximately 800 MB

where
      100           = the message rate a second
      (1 + 1.3) KB  = the amount of data logged for
                      each 1 KB of persistent messages
```

Consider an overall target recovery time of 75 minutes. If you have allowed 15 minutes to react to the problem and restore the page set backup copy, queue manager recovery and restart must then complete within 60 minutes (3600 seconds) applying formula (A). Assuming that all required log data is on RVA2-T82 DASD, which has a recovery rate of approximately 2.7 MB a second, this necessitates a page set backup frequency of at least every:

```
  3600 seconds * 2.7 MB a second / 800 MB an hour  =  12.15 hours
```

If your WebSphere MQ application day lasts approximately 12 hours, one backup each day is appropriate. However, if the application day lasts 24 hours, two backups each day is more appropriate.

Another example might be a production system in which all the messages are for request-reply applications (that is, a persistent message is received on a receiver channel and a persistent reply message is generated and sent down a sender channel).

In this example, the achieved batch size is one, and so there is one batch for every message. If there are 50 request replies a second, the total load is 100 persistent messages a second. If each message is 1 KB in length, the amount of data logged each hour is approximately:

```
50((2 * (1+1.3) KB) + 1.4 KB + 2.5 KB) * 3600 = approximately 1500 MB

where:
 50                 = the message pair rate a second
 (2 * (1 + 1.3) KB) = the amount of data logged for each message pair
 1.4 KB             = the overhead for each batch of messages
                      received by each channel
 2.5 KB             = the overhead for each batch of messages sent
                      by each channel
```

To achieve the queue manager recovery and restart within 30 minutes (1800 seconds), again assuming that all required log data is on RVA2-T82 DASD, this requires that page set backup is carried out at least every:

```
    1800 seconds * 2.7 MB a second / 1500 MB an hour = 3.24 hours
```

Periodic review of backup frequency

You are recommended to monitor your WebSphere MQ log usage in terms of MB an hour. You should perform this check periodically and amend your page set backup frequency if necessary.

Backup and recovery with DFHSM

The data facility hierarchical storage manager (DFHSM) does automatic space-availability and data-availability management among storage devices in your system. If you use it, you need to know that it moves data to and from the WebSphere MQ storage automatically.

DFHSM manages your DASD space efficiently by moving data sets that have not been used recently to alternate storage. It also makes your data available for recovery by automatically copying new or changed data sets to tape or DASD backup volumes. It can delete data sets, or move them to another device. Its operations occur daily, at a specified time, and allow for keeping a data set for a predetermined period before deleting or moving it.

You can also perform all DFHSM operations manually. The *Data Facility Hierarchical Storage Manager User's Guide* explains how to use the DFHSM commands. If you use DFHSM with WebSphere MQ, note that DFHSM does the following:

- Uses cataloged data sets.
- Operates on page sets and logs.
- Supports VSAM data sets.

Recovery and CICS

The recovery of CICS resources is not affected by the presence of WebSphere MQ. CICS recognizes WebSphere MQ as a non-CICS resource (or external resource manager), and includes WebSphere MQ as a participant in any syncpoint coordination requests using the CICS resource manager interface (RMI). For more information about CICS recovery, see the *CICS Recovery and Restart Guide*. For information about the CICS resource manager interface, see the *CICS Customization Guide*.

Recovery and IMS

IMS recognizes WebSphere MQ as an external subsystem and as a participant in syncpoint coordination. IMS recovery for external subsystem resources is described in the *IMS Customization Guide*.

Preparing for recovery on an alternative site

In the case of a total loss of a WebSphere MQ computing center, you can recover on another WebSphere MQ system at a recovery site. To do this, you must regularly back up the page sets and the logs. As with all data recovery operations, the objectives of disaster recovery are to lose as little data, workload processing (updates), and time as possible.

At the recovery site:

- The recovery WebSphere MQ queue manager **must** have the same name as the lost queue manager.
- The system parameter module used on the recovery queue manager should contain the same parameters as the lost queue manager.

The process for disaster recovery is described in the WebSphere MQ for z/OS System Administration Guide.

Example of queue manager backup activity

When you plan your queue manager backup strategy, a key consideration is retention of the correct amount of log data. The WebSphere MQ for z/OS System Setup Guide describes how to determine which log data sets are required, by reference to the system recovery RBA of the queue manager. WebSphere MQ determines the system recovery RBA using information about the following:

- Currently active units of work.
- Page set updates that have not yet been flushed from the buffer pools to disk.
- CF structure backups, and whether this queue manager's log contains information required in any recovery operation using them.

You must retain sufficient log data to be able to perform media recovery. Whilst the system recovery RBA increases over time, the amount of log data that must be retained only decreases when subsequent backups are taken. CF structure backups are managed by WebSphere MQ, and so are taken into account when reporting the system recovery RBA. This means that in practice, the amount of log data that must be retained only reduces when page set backups are taken.

Figure 23 on page 134 shows an example of the backup activity on a queue manager that is a member of a queue-sharing group, how the recovery RBA varies

with each backup, and how that affects the amount of log data that must be retained. In the example the queue manager uses local and shared resources: page sets, and two CF structures, STRUCTURE1 and STRUCTURE2.

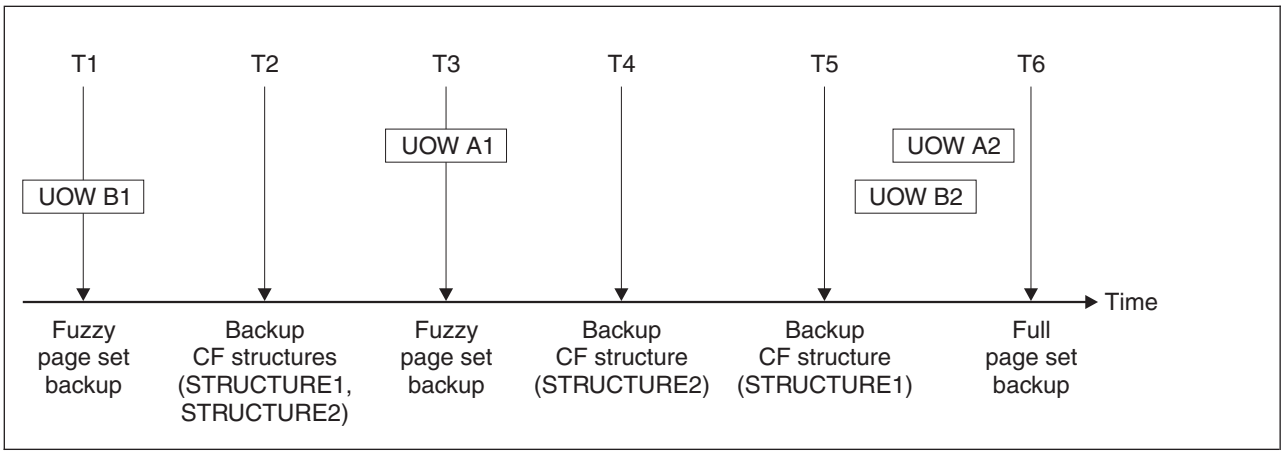


Figure 23. Example of queue manager backup activity

This is what happens at each point in time:

Point in time T1

A fuzzy backup is created of your page sets, as described in WebSphere MQ for z/OS System Administration Guide.

The system recovery RBA of the queue manager is the lowest of the following:

- The recovery RBAs of the page sets being backed up at this point.
- The lowest recovery RBA required to recover the CF application structures. This relates to the recovery of backups of STRUCTURE1 and STRUCTURE2 created earlier.
- The recovery RBA for the oldest currently active unit of work within the queue manager (UOWB1).

The system recovery RBA for this point in time is given by messages issued by the DISPLAY USAGE command, which is part of the fuzzy backup process.

Point in time T2

Backups of the CF structures are created. CF structure STRUCTURE1 is backed up first, followed by STRUCTURE2.

The amount of log data that must be retained is unchanged, because the same data as determined from the system recovery RBA at T1 is still required to recover using the page set backups taken at T1.

Point in time T3

Another fuzzy backup is created.

The system recovery RBA of the queue manager is the lowest of the following:

- The recovery RBAs of the page sets being backed up at this point.
- The lowest recovery RBA required to recover CF structure STRUCTURE1, because STRUCTURE1 was backed up before STRUCTURE2.
- The recovery RBA for the oldest currently active unit of work within the queue manager (UOWA1).

The system recovery RBA for this point in time is given by messages issued by the DISPLAY USAGE command, which is part of the fuzzy backup process.

You can now reduce the log data retained, as determined by this new system recovery RBA.

Point in time T4

A backup is taken of CF structure STRUCTURE2. The recovery RBA for the recovery of the oldest required CF structure backup relates to the backup of CF structure STRUCTURE1, which was backed up at time T2.

The creation of this CF structure backup has no impact on the amount of log data that must be retained.

Point in time T5

A backup is taken of CF structure STRUCTURE1. The recovery RBA for recovery of the oldest required CF structure backup now relates to recovery of CF structure STRUCTURE2, which was backed up at time T4.

The creation of this CF structure backup has no impact on amount of log data that must be retained.

Point in time T6

A full backup is taken of your page sets as described in WebSphere MQ for z/OS System Administration Guide.

The system recovery RBA of the queue manager is the lowest of the following:

- The recovery RBAs of the page sets being backed up at this point.
- The lowest recovery RBA required to recover the CF structures. This relates to recovery of CF structure STRUCTURE2.
- The recovery RBA for the oldest currently active unit of work within the queue manager. In this case, there are no current units of work.

The system recovery RBA for this point in time is given by messages issued by the DISPLAY USAGE command, which is part of the full backup process.

Again, the log data retained can be reduced, because the system recovery RBA associated with the full backup is more recent.

Chapter 5. Planning to install WebSphere MQ

WebSphere MQ Prerequisites

This chapter discusses the prerequisite products that you need to install before you can install and use WebSphere MQ for z/OS.

Hardware requirements

WebSphere MQ runs on any IBM zSeries or S/390® processor that is capable of running the required level of z/OS and that has enough storage to meet the combined requirements of the programming prerequisites, WebSphere MQ, the access methods, and the application programs.

Software requirements

This section lists the software requirements for WebSphere MQ for z/OS.

You can use one of the packaged offerings of z/OS from IBM (for example, ServerPac). This includes most of the products that you need to run WebSphere MQ for z/OS, and the integrated products have been verified by IBM.

The list of elements included in z/OS, and the recommended levels of nonexclusive elements are described in the *z/OS and z/OSe Planning For Installation* manual.

WebSphere MQ for z/OS, V7.0 requires z/OS Version 1.8 or later, together with the products included in z/OS. These include:

- C/C++

  To take advantage of the new WebSphere MQ V7.0 functionality from a C application, you must use a compiler that supports the *long long* integer type.
- optionally Cryptographic Services System SSL (if you want to use the Secure Sockets Layer (SSL) for channel security)
- optionally Cryptographic Services Security Level 3 (if you want to use SSL for channel security with US encryption strengths)
- DFSMS/DFP
- High Level Assembler
- ICSS
- ISPF
- JES
- Language Environment®
- Security Server
- SMP/E
- Optionally TCP/IP (see “Communications protocol and distributed queuing” on page 139)
- TSO/E
- UNIX System Services
- VTAM

The following list gives the minimum levels required for the optional products that are not included in z/OS. You can use any versions of COBOL and PL/I compilers that can generate calls to WebSphere MQ conforming to the standard operating system linkage conventions described in the *z/OS MVS Assembler Services Guide*. You might not need all the following:

- CICS Transaction Server Version 2.3
- IMS, Version 8.1
- IBM SDK for z/OS, Java 2 Technology Edition, Version 1.4.2

Additional requirements for some features

Some of the features of WebSphere MQ for z/OS, V7.0 have additional requirements.

These are listed below:

Queue-sharing groups

- z/OS V1.8 with Resource Recovery Services (RRS) activated and Coupling Facility Level 9
- DB2 Version 8.1.

Topic security

To implement TOPIC security or exploit mixed case profiles for other MQ objects

- RACF APAR 0A23043 for z/OS 1.8
- z/OS 1.9 or later

Non-IBM products

If you choose to use Unicenter TCPaccess Communication Server (formerly called SOLVE:TCPaccess) from Computer Associates instead of IBM's TCP/IP, the recommended minimum level is Version 5.2.

Clients

For WebSphere MQ for z/OS to support clients you need to install the client/server support code that is provided by the Client Attachment feature of WebSphere MQ for z/OS. This must be the same version as WebSphere MQ.

Delivery

WebSphere MQ for z/OS is supplied on 3480 cartridge only.

One cartridge contains the product code together with the following language features; U.S. English (mixed case), U.S. English (uppercase), Chinese, and Japanese. The optional Client Attachment feature is supplied on a separate cartridge.

Making WebSphere MQ available

This chapter gives an overview of what you need to do to make WebSphere MQ available to application programmers, and their applications. It contains the following sections:

Installing WebSphere MQ for z/OS

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

SMP/E, used for installation on the z/OS platform, validates the service levels and prerequisite and corequisite products, and maintains the SMP/E history records to record the installation of WebSphere MQ for z/OS. It loads the WebSphere MQ for z/OS libraries and checks that the loads have been successful. You then have to customize the product to your own requirements.

Before you install and customize WebSphere MQ for z/OS, you must decide the following:

- Whether you are going to install one of the optional national language features.
- Which communications protocol and distributed queuing facility you are going to use.
- What your naming convention for WebSphere MQ objects will be.
- What command prefix string (CPF) you are going to use for each queue manager.

You also need to plan how much storage you require in your z/OS system to accommodate WebSphere MQ; "Planning your storage and performance requirements" on page 101 helps you plan the amount of storage required.

National language support

You can choose one of the following national languages for the WebSphere MQ operator messages and the WebSphere MQ operations and control panels (including the character sets used). Each language is identified by one of the following language letters:

C Simplified Chinese

E U.S. English (mixed case)

K Japanese

U U.S. English (uppercase)

The samples, WebSphere MQ commands, and utility control statements are available only in mixed case U.S. English.

Communications protocol and distributed queuing

The distributed queuing facility provided with the base product feature of WebSphere MQ can either use APPC (LU 6.2), TCP/IP from IBM, or Unicenter TCPaccess Communication Server (formerly called SOLVE:TCPaccess). The distributed queuing facility is also known as the channel initiator and the mover.

If you want to use clients, the Client Attachment feature is needed as well (but you can administer clients without it).

You must perform the following tasks to enable distributed queuing:

- Choose which communications interface to use. This can be one of the following:
  - APPC (LU 6.2)
  - TCP/IP
- Customize the distributed queuing facility and define the WebSphere MQ objects required.
- Define access security.
- Set up your communications. This includes setting up your TCPIP.DATA data set if you are using TCP/IP, LU names and side information if you are using APPC. This is described in the WebSphere MQ Intercommunication manual.

Naming conventions

It is advisable to establish a set of naming conventions when planning your WebSphere MQ systems. The names you choose will probably be used on different platforms, so you should follow the convention for WebSphere MQ, not for the particular platform.

WebSphere MQ allows both uppercase and lowercase letters in names, and the names are case sensitive. However, some z/OS consoles fold names to uppercase, so do not use lowercase letters for names unless you are sure that this will not happen.

You can also use numeric characters and the period (.), forward slash (/), underscore (_) and percent (%) characters. The percent sign is a special character to Security Server (previously known as RACF), so do not use it in names if you are using Security Server as your External Security Manager. Do not use leading or trailing underscore characters if you are planning to use the Operations and Control panels.

Rules for naming WebSphere MQ objects are described in the WebSphere MQ Script (MQSC) Command Reference manual.

Choosing names for queue managers and queue-sharing groups:

Follow these guidelines to ensure that you use unique names for your queue managers and queue-sharing groups.

Each queue manager and queue-sharing group within a network must have a unique name. Do not use the same name for a queue manager and a queue-sharing group. On z/OS the names of queue managers and queue-sharing groups can be up to four characters long. Each DB2 system and data-sharing group within the network must also have a unique name.

The names of queue manager and queue-sharing groups can use only uppercase alphabetic characters, numeric characters, and dollar sign ($), number sign (#) or at sign (@); they must not start with a numeric character. Queue-sharing group names that are less than four characters long are padded internally with at signs, so do not use names ending in the at sign.

The queue manager name is the same as the z/OS subsystem name. You might identify each subsystem as a queue manager by giving it the name `QM`*`xx`* (where *xx* is a unique identifier), or you might choose a naming convention similar to `ADDX`, where `A` signifies the geographic area, `DD` signifies the company division, and `X` is a unique identifier.

You might want to use your naming convention to distinguish between queue managers and queue-sharing groups. For example, you might identify each queue-sharing group by giving it the name QG*xx* (where *xx* is the unique identifier).

Choosing names for objects:

Queues, processes, namelists, and clusters can have names up to 48 characters long. Channels can have names up to 20 characters long and storage classes can have names up to 8 characters long.

If possible, choose meaningful names within any constraints of your local conventions. Any structure or hierarchy within names is ignored by WebSphere MQ, however, hierarchical names can be useful for system management. You can also specify a description of the object when you define it to give more information about its purpose.

Each object must have a unique name within its object type. However, each object type has a separate name space, so you can define objects of different types with the same name. For example, if a queue has an associated process definition, it is a good idea to give the queue and the process the same name. It is also a good idea to give a transmission queue the same name as its destination queue manager.

You could also use the naming convention to identify whether the object definition is private or a global. For example, you could call a namelist `project_group.global` to indicate that the definition is stored on the shared repository.

Application queues:

Choosing names that describe the function of each queue helps you to manage these queues more easily. For example, you could call a queue for inquiries about the company payroll `payroll_inquiry`. The reply-to queue for responses to the inquiries could be called `payroll_inquiry_reply`.

You can use a prefix to group related queues. This means that you can specify groups of queues for administration tasks like managing security and using the dead-letter queue handler. For example, all the queues that belong to the payroll application could be prefixed by `payroll_`. You can then define a single security profile to protect all queues with names beginning with this prefix.

You can also use your naming convention to indicate that a queue is a shared queue. For example, if the payroll inquiry queue mentioned above was a shared queue, you could call it `payroll_inquiry.shared`.

Storage classes and Coupling Facility structures:

The character set you can use when naming storage classes and Coupling Facility structures is limited to uppercase alphabetic and numeric characters. You should be systematic when choosing names for these objects.

Storage class names can be up to 8 characters long, and must begin with an alphabetic character. You will probably not define many storage classes, so a simple name is sufficient. For example, a storage class for IMS bridge queues could be called `IMS`.

Coupling Facility structure names can be up to 12 characters long, and must begin with an alphabetic character. You could use the name to indicate something about

the shared queues associated with the Coupling Facility structure (that they all belong to one suite of applications for example). Remember that in the Coupling Facility itself, the structure names are the WebSphere MQ name prefixed by the name of the queue-sharing group (padded to four characters with @ symbols).

Choosing names for channels:

To help you manage channels, it is a good idea if the channel name includes the names of the source and target queue managers. For example, a channel transmitting messages from a queue manager called QM27 to a queue manager called QM11 might be called `QM27/QM11`.

If your network supports both TCP and SNA, you might also want to include the transport type in the channel name, for example `QM27/QM11_TCP`. You could also indicate whether the channel is a shared channel, for example `QM27/QM11_TCP.shared`.

Remember that channel names cannot be longer than 20 characters. If you are communicating with a queue manager on a different platform, where the name of the queue manager might contain more than 4 characters, you might not be able to include the whole name in the name of the channel.

Using command prefix strings

Each instance of WebSphere MQ that you install must have its own *command prefix* string (CPF). You use the CPF to identify the z/OS subsystem that commands are intended for. It also identifies the z/OS subsystem from which messages sent to the console originate.

You can issue all MQSC commands from an authorized console by inserting the CPF before the command. If you enter commands through the system command input queue (for example, using CSQUTIL), or use the WebSphere MQ operations and control panels, you do not use the CPF.

To start a subsystem called CSQ1 whose CPF is '`+CSQ1`', issue the command `+CSQ1 START QMGR` from the operator console (the space between the CPF and the command is optional).

The CPF also identifies the subsystem that is returning operator messages. The following example shows `+CSQ1` as the CPF between the message number and the message text.

```
CSQ9022I +CSQ1 CSQNCDSP ' DISPLAY CMDSERV' NORMAL COMPLETION
```

See the WebSphere MQ for z/OS System Setup Guide for information about defining command prefix strings.

Customizing WebSphere MQ and its adapters

WebSphere MQ requires some customization after installation to meet the individual and special requirements of your system, and to use your system resources in the most effective way. These are the tasks that you must perform when you customize your system:

1. Identify the z/OS system parameters

2. APF authorize the WebSphere MQ load libraries
3. Update the z/OS link list and LPA
4. Update the z/OS program properties table
5. Define the WebSphere MQ subsystem to z/OS
6. Create procedures for the WebSphere MQ subsystem
7. Create procedures for the channel initiator
8. Set up the DB2 environment
9. Set up the Coupling Facility
10. Implement your ESM security controls
11. Update SYS1.PARMLIB members
12. Customize the initialization input data sets
13. Create the bootstrap and log data sets
14. Define your page sets
15. Add the WebSphere MQ entries to the DB2 data-sharing group
16. Tailor your system parameter module
17. Tailor the channel initiator parameter module
18. Set up Batch, TSO, and RRS adapters
19. Set up the operations and control panels
20. Include the WebSphere MQ dump formatting member
21. Suppress information messages

These tasks are described in detail in the WebSphere MQ for z/OS System Setup Guide.

Using queue-sharing groups

If you want to use queue-sharing groups, you do not have to set them up when you install WebSphere MQ, you can do this at any time.

See the migration section in the WebSphere MQ for z/OS System Setup Guide for details of how to set up a new queue-sharing group. See the WebSphere MQ for z/OS System Administration Guide for details of how to manage your queue-sharing groups when you have set them up.

Verifying your installation of WebSphere MQ for z/OS

After the installation and customization has been completed, you can use the installation verification programs (IVPs) supplied with WebSphere MQ to verify that the installation has been completed successfully. The IVPs supplied are assembler language programs and you should run them after you have customized WebSphere MQ to suit your needs. They are described in the WebSphere MQ for z/OS System Setup Guide.

What's changed in WebSphere MQ Version 7.0

This chapter provides a short description of new and changed features. It also provides an overview of what to consider when migrating from a previous version of WebSphere MQ for z/OS. It contains the following sections:

- "What's new in WebSphere MQ for z/OS" on page 144
- "Migration from previous versions" on page 151

What's new in WebSphere MQ for z/OS

This section describes the new function that has been added in WebSphere MQ for z/OS, V7.0.

Security

Refresh SSL cache:

It is now possible to refresh the cached view of the SSL Key Repository by issuing the REFRESH SECURITY command with the option TYPE(SSL). For more information, see the *WebSphere MQ Script (MQSC) Command Reference*.

Reset SSL:

Parameters SSLKEYDA, SSLKEYTI, and SSLRKEYS are introduced to display the date and time of the previous successful key reset, and the number of successful key resets since channel start. The DISPLAY CHSTATUS command has been updated. For more information, see the *WebSphere MQ Script (MQSC) Command Reference*.

PCF commands:

Most MQSC commands now have PCF equivalents. These generally use the same security profiles but there are some which need new profiles, for further information, see the *WebSphere MQ for z/OS System Setup Guide.*

MQSC commands:

The MQSC command DISPLAY DQM has been renamed DISPLAY CHINIT, although the old command still works as a synonym. If you have a security profile for the old command, you should rename it.

There are new and extended MQSC commands DISPLAY CONN, ALTER BUFFPOOL, DELETE BUFFPOOL, ALTER PSID, DELETE PSID, DEFINE PSID, DEFINE LOG, for which you need security profiles.

RACF classes:

You can now use mixed case RACF profile support, which allows you to use mixed case resource names and define WebSphere MQ RACF profiles to protect them.

Without the use of mixed case RACF profiles, you can still use mixed case resource names in WebSphere MQ for z/OS; however, these resource names can only be protected by generic RACF profiles in the uppercase WebSphere MQ classes. When using mixed case WebSphere MQ RACF profile support you can provide a more granular level of protection by defining WebSphere MQ RACF profiles in the mixed case WebSphere MQ classes.

There are some profiles, or parts of profiles, that remain uppercase only as the values are provided by WebSphere MQ. These are:

- Switch profiles.
- All high-level qualifiers (HLQ) including subsystem and Queue-Sharing Group identifiers.
- Profiles for SYSTEM objects.

- Profiles for Default objects.
- The MQCMDS class, so all command profiles are uppercase only.
- The MQCONN class, so all connection profiles are uppercase only.
- RESLEVEL profiles.
- The `'object'` qualification in command resource profiles; for example, `hlq.QUEUE.queuename`. The resource name only is mixed case.
- Dynamic queue profiles `hlq.CSQOREXX.*` , `hlq.CSQUTIL.*`, and `CSQXCMD.*`.
- The `'CONTEXT'` part of the `hlq.CONTEXT.resourcename`.

Page sets and buffer pools

Dynamically add or remove buffers in an existing buffer pool:

WebSphere MQ allows you to dynamically add more buffers to a buffer pool, if the pool is too small. The new command is ALTER BUFFPOOL. See the WebSphere MQ Script (MQSC) Command Reference for more information about this command.

Large page sets:

Large page sets of greater than 4GB in size provide extra capacity for messages, for example if the network stops. These can now be created on WebSphere MQ, if required. Because the throughput of the system is not affected, the existing logs and buffer pools provide adequate support. For more information, see “Storage management” on page 20.

Page set expansion:

Page set expansion has been improved. A page set is expanded before it becomes full, and the expansion is performed asynchronously from the MQPUT command. The new secondary extent of the page set increases in line with the size of the page set. Page set definitions that specify no secondary extent size are still expandable, whereas previously the expansion failed. The DEFINE PSID and ALTER PSID commands have been extended to support this feature.

For information about enabling page set expansion, see WebSphere MQ for z/OS Concepts and Planning Guide.

For information about the DEFINE PSID and ALTER PSID commands, see WebSphere MQ Script (MQSC) Command Reference.

Dynamically add log data sets to an active queue manager

Log data in WebSphere MQ is written to a ring of active logs. If all the logs in the ring become filled up, the queue manager is unable to perform logging, work stops and an application failure occurs. To avoid this scenario, it is now possible to add additional log data sets to the active queue manager. This allows more time for fixing the underlying log offload problem. It also allows additional time for message processing to reach the end of a critical period, if necessary.

In this context, a new MQSC command DEFINE LOG allows you to specify the name of a new VSAM data set, and the number of an active log ring. For additional information, see “Logging” on page 25. See the WebSphere MQ Script (MQSC) Command Reference for details of this command.

Display connection information (DISPLAY CONN command)

A new command DISPLAY CONN displays connection information for applications that are connected to the queue manager. The command displays details of objects, and transactional information for a particular connection. The log time information returned by this command indicates if applications have a long running unit of work.

See the WebSphere MQ Script (MQSC) Command Reference for details of this command.

Commands

Programmable command format (PCF) support:

Support for programmable command format (PCF).

WebSphere MQ for z/OS supports the use of PCF commands. The commands must be put to the SYSTEM.COMMAND.INPUT queue, and must have a PCF message type of MQCFT_COMMAND_XR and a format of MQFMT_ADMIN.

For details of the PCF commands, see the *Programmable Commands Format and Administration Interface* guide.

Command tracking notification:

To provide additional support during problem diagnosis, a new event queue SYSTEM.ADMIN.COMMAND.EVENT has been implemented, together with a new queue manager attribute that controls the new queue. Messages are automatically written to the new queue whenever commands are issued.

Filtering on the DISPLAY commands:

A new keyword, WHERE, is provided for the Script (MQSC) DISPLAY commands, and the PCF Display commands. This allows you to filter the information displayed by one of the attributes of objects.

For information about using the new keyword with the Script (MQSC) DISPLAY commands, see the WebSphere MQ Script (MQSC) Command Reference.

For information about using the new keyword with the PCF Display commands, see the *Programmable Commands and Administration Interface* book.

Queue sharing group improvements

Support for large messages on shared queues:

WebSphere MQ can now handle large messages up to 100 MB on shared queues. A new CFLEVEL(4) has been introduced to support this function. A prerequisite for this is DB2 7.1 or later with the fix for APAR PQ71179 is also required.

Availability:

If you are using queue-sharing groups and the administration structure fails, the queue managers that are active in the queue-sharing group no longer terminate. Instead, work is suspended, the structure is automatically reallocated and rebuilt, and then the work continues.

For more information about what happens when an administration structure fails, see WebSphere MQ for z/OS Concepts and Planning Guide.

Queue manager removal:

Before attempting to remove a queue manager from a queue sharing group, it is necessary to check whether the queue manager has an interest in a CF structure backup. This information can now be viewed using the DISPLAY CFSTATUS command.

Queue-sharing group utility:

A new VERIFY QSG function has been added to CSQ5PQSG, the queue-sharing group utility. You can use the VERIFY QSG function to validate the consistency of the DB2 object definitions for queue managers, structures, and shared queues.

For information about CSQ5PQSG, see WebSphere MQ for z/OS System Administration Guide.

Improved cluster workload management

New attributes have been added to queues, queue managers, and cluster channels to allow users more control over how workload management is performed within clusters.

For information about the new attributes, see WebSphere MQ Script (MQSC) Command Reference.

For information about workload management within clusters, see WebSphere MQ Queue Manager Clusters.

Improvements to SSL support

You can now renegotiate a new secret key periodically while a channel is running, and the connection is not interrupted. A secret key is generated from the random text sent as a part of the SSL handshake. If the secret key is in existence for a long period of time it could be discovered, and all data encrypted with the same secret key can be deciphered.

Two additional CipherSpecs have been added:
TLS_RSA_WITH_AES_128_CBC_SHA and TLS_RSA_WITH_AES_256_CBC_SHA.

The limitation on the number of AUTHINFO objects that can be used from the SSLCRLNL has been increased from one to ten. This ensures continuity of service if one or more LDAP servers fail.

For information about SSL, see WebSphere MQ Security.

Integrated accounting and statistics data formatter

The MP1B accounting and statistics data formatter SupportPac has been integrated into WebSphere MQ for z/OS. The function allows you to format and print the accounting and statistics data within the SMF records produced by WebSphere MQ for z/OS.

For information about this function, see *WebSphere MQ for z/OS System Administration Guide*.

Online monitoring

You can now view current monitoring and status data for a channel or queue using the DISPLAY CHSTATUS and the DISPLAY QSTATUS commands, which have been extended to support online monitoring. The monitoring information can be used to help gauge the performance and health of the system.

Monitoring is controlled by new queue manager, queue and channel attributes.

For information about measuring system effectiveness, see the *Monitoring WebSphere MQ* book.

For information about the DISPLAY commands, see WebSphere MQ Script (MQSC) Command Reference.

Channels

Compression of channel data:

Compression of channel data reduces the amount of network traffic and can therefore improve the performance of channels. Channel data compression is enabled through the use of channel send and receive exits. New versions of the MQCD and MQCXP have been added, and new attributes have been added to the DEFINE CHANNEL and ALTER CHANNEL commands, to support channel compression.

For information about changes to MQCD and MQCXP, see WebSphere MQ Intercommunication.

For information about the DEFINE CHANNEL and ALTER CHANNEL commands, see WebSphere MQ Script (MQSC) Command Reference.

Channel event control:

It is now possible to enable or disable channel event messages, IMS Bridge event messages and SSL event (these are the messages that are put to the SYSTEM.ADMIN.CHANNEL.EVENT queue). Previously the only way to do this was to delete the queue itself.

It is also now possible to collect only those event messages that are of interest (that is, event messages related to channel errors rather than those related to the normal starting and stopping of channels).

To support this feature, changes have been implemented to the commands ALTER QMGR and DISPLAY QMGR.

Message retry on channels:

Message retry is now available using the MRRTY, MRTMR, MREXIT and MRDATA channel attributes.

Migrated channels from previous releases have the default value MRRTY(0) so there is no message retry, and new channels have the default value MRRTY(10).

Note that you can change the message retry default settings by altering the SYSTEM.DEF* channels.

Heartbeat for clients:

When an MQI channel is waiting, particularly for a GET with WAIT, it is useful to be able to poll the channel to verify that the two MCAs are still active.

In this context, heartbeat interval is now supported for client-connection channels.

To support this feature, changes have been implemented for the ALTER CHANNEL and DEFINE CHANNEL MQSC commands.

Support for IPv6:

IPv6 (Internet Protocol Version 6) is designed by the IETF to replace the current version Internet Protocol, Version 4 (IPv4). IPv6 fixes a number of problems in IPv4, such as the limited number of available addresses, as well as adding improvements in areas such as routing and automatic configuration of networks . IPv6 is expected to gradually replace IPv4, with the two protocols co-existing for a number of years during a transition period. WebSphere MQ for z/OS offers support for intercommunication using IPv6, and continues to support the use of the IPv4 protocol.

To support IPv6, a new attribute, IPADDR has been added to the ALTER QMGR and DISPLAY QMGR commands. For details of these commands, see WebSphere MQ Script (MQSC) Command Reference.

For information about using IPv6, see the *WebSphere MQ Internet Protocol Version 6 (IPv6) Migration* publication.

Channel initiator

Dynamic changing of parameters:

Channel initiator parameters were originally implemented using assembler load modules, but have now been moved to be queue manager attributes. This means it is now possible to dynamically modify them using the ALTER QMGR command and view them using the DISPLAY QMGR command. A utility for migrating existing parameter load modules has also been provided.

To summarize:

- The ALTER QMGR command has been extended so that it now supports all the parameters previously supplied in the CSQ6CHIP macro.
- A migration facility is now provided in CSQUTIL for these parameters. This simplifies the migration process by allowing the load module to be migrated when the queue manager is not running, before the queue manager has been migrated to WebSphere MQ V6
- The PARM parameter has been removed from the START CHINIT command on z/OS.
- The DISPLAY QMGR command has been extended so that it now presents the values of these parameters.

For information about the commands, see WebSphere MQ Script (MQSC) Command Reference or WebSphere MQ Programmable Command Formats and Administration Interface.

For information about how the channel initiator is affected by this feature, see WebSphere MQ for z/OS System Setup Guide.

TCP/IP interface:

Only one type of TCP/IP interface is now supported, which is that formerly specified by TCPTYPE=OESOCKET in the channel initiator parameters.

Multiple TCP/IP stacks:

WebSphere MQ can now use more than one TCP/IP stack for both inbound as well as outbound communication. This provides greater flexibility when configuring WebSphere MQ on a system with than one network interface. If more than one network interface is involved, it is now possible to use more than one of them for your WebSphere MQ channels. This means it is no longer necessary to choose just one of these for all outbound channels, or install a second queue manager on the same system.

For more information see *WebSphere MQ Intercommunications*, and the *WebSphere MQ Script (MQSC) Command Reference*.

CICS bridge improvements

There have been several improvements to the CICS bridge to allow work to improve availability and scalability by allowing work to be distributed across multiple CICS regions.

Multiple bridge monitors:

The Bridge monitor task for a bridge queue can be run in multiple CICS regions connected to queue managers allowing, with shared queues, the work to be distributed across a CICSplex. To support user verification by RACF passtickets across multiple regions a new PASSTKTA applid keyword has been added to monitor start up.

Note: Existing 3270 bridge applications may need to be updated to propagate the MQCIH RemoteSysId field returned by the first transaction of a pseudo-conversation into the request messages of subsequent transactions. Multiple bridge monitors cannot be used from CICS regions attached to pre-V6 queue managers in a queue sharing group.

Transaction routing:

User applications started by the bridge monitor no longer need to be run in the same CICS region as the bridge monitor this can be achieved by either CICS transaction routing facilities (when using CICS Transaction Server V2.2 or later) or MQ transaction routing by specifying the desired CICS system id in the MQCIH RemoteSysId field. When using CICS routing the target CICS region does not have to be connected to a queue manager while with MQ routing the target region must be connected to a queue manager that has access to the bridge queue.

Pseudoconversational programming model:

The normal CICS programming model is pseudo-conversational with individual transactions linked to form a conversational flow and in the past the MQ messages sent to the bridge had to follow the same model with the first message for each transaction being a new session message. Now, when using CICS Transaction

Server V2.2 or later, it is possible to include all of the MQ messages for multiple transactions within the same bridge session which reduces monitoring overheads and improves performance.

Additional headers tolerated:

MQ headers with format names starting MQH and standard header chaining fields can be included in MQ bridge messages prior to the MQCIH header. They are not processed by the bridge but are returned unchanged in the bridge reply messages. Applications could use these headers to contain state data that they wish to flow with the message.

Expiration time control:

To control the expiration time of reply messages application can specify either MQRO_PASS_DISCARD_AND_EXPIRY in the MQMD Report field or MQCIH_PASS_EXPIRATION in the MQCIH flags field. If either option is specified the reply message will have the remaining expiry time from the request message rather than unlimited expiry.

Failed messages to backout requeue queue:

If a backout requeue queue is defined for the bridge queue it will be used for messages that have been backed out more than the backout threshold instead of being written to the dead-letter queue. If the backout queue doesn't exist or can't be written to then the dead-letter queue will be used unless the MQRO_DISCARD_MSG report option has been specified.

Operator messages destination control:

Messages generated by the bridge can now be directed to the CICS master terminal (CSMT), CICS job log, or both.

Software prerequisites

Software prerequisites have changed. See"Software requirements" on page 137 for details of what you need.

Removed features

The Application Messaging Interface (AMI) is no longer part of WebSphere MQ for z/OS. It is now available as the 'MA0F MQSeries Application Messaging Interface' SupportPac. See www.ibm.com/software/integration/support/supportpacs/ for a list of all the available IBM SupportPacs.

The CICS Mover is no longer supported.

The Internet Gateway is no longer supported.

Migration from previous versions

Migration from earlier versions

If you are migrating to WebSphere MQ Version 7 from a version earlier than Version 6 (that is, WebSphere MQ Version 5.3.1 or earlier), you must perform a two-stage migration: first migrate to WebSphere MQ Version 6, then migrate to WebSphere MQ Version 7.

For more information, see *WebSphere MQ Migration Information*.

Additional steps related to migration from Version 5.3

The following additional factors must be taken into consideration when migrating from Version 5.3:

- Channel initiator — parameters and trace commands have changed
- Libraries — one library is no longer used

For more information, see the *WebSphere MQ for z/OS System Setup Guide*.

Migration to full function WebSphere MQ

The following factors must be taken into consideration when migrating from the reduced function form of WebSphere MQ Version 5.3.1 supplied with WebSphere Application Server:

- Queue managers — may require customization
- Initialization input data set — may require customization
- System parameter module — may require customization
- Channel initiator parameter — may customization
- Customizing for CICS — may be required
- Customizing for IMS — may be required
- Queue-sharing groups — may require setting up (if applicable)
- Client Attachment feature — may require installing (if applicable)

For more information, see the *WebSphere MQ for z/OS System Setup Guide*.

Reverting to a previous version

After you have migrated to WebSphere MQ for z/OS, Version 7, you can revert to using a previous version of WebSphere MQ for z/OS if there are exceptional circumstances. However to do this, you must apply a PTF to the previous version. This is described in *WebSphere MQ Migration Information*

Coexistence with earlier versions of WebSphere MQ

A number of factors must be taken into consideration when coexisting with earlier versions of WebSphere MQ in a queue-sharing group.

For more information, see *WebSphere MQ Migration Information*.

Chapter 6. Macros intended for customer use

The macros identified in this appendix are provided as programming interfaces for customers in support of features that are specific to WebSphere MQ for z/OS.

The 'C' include files, COBOL copy files, PL/I include files and assembler macros that are provided as programming interfaces for customers in support of features that apply across many WebSphere MQ platforms are described in the *WebSphere MQ Constants* book.

Note: Do not use as programming interfaces any WebSphere MQ macros other than those identified in this appendix or in the *WebSphere MQ Constants* book.

General-use programming interface macros

The following assembler macros are provided to enable you to write programs that use the services of WebSphere MQ. The macros are supplied in library thlqual.SCSQMACS.

- CMQXCALA
- CMQXCFBA
- CMQXCFCA
- CMQXCFLA
- CMQXCDFA
- CMQXCINA
- CMQXCVCA

Product-sensitive programming interface macros

The following assembler macros are provided to enable you to write programs that use the services of WebSphere MQ. The macros are supplied in library thlqual.SCSQMACS.

CSQBDEF	CSQDQEST	CSQDQIST	CSQDQJST
CSQDQLST	CSQDQMAC	CSQDQMST	CSQDQPST
CSQDQSST	CSQDQWHC	CSQDQWHS	CSQDQ5ST
CSQDWQ	CSQDWTAS	CSQQDEFX	CSQQLITX

Chapter 7. Measured usage license charges with WebSphere MQ for z/OS

Measured Usage License Charges (MULC) is a particular way of charging you for an IBM product that runs on a z/OS system, based on how much use you make of the product. To determine the product usage, the z/OS system records the amount of processor time that is used by the product when it executes.

z/OS can measure how much processing time is spent in doing work on behalf of the WebSphere MQ queue manager that is handling MQI calls, executing MQSC commands, or performing some other action to support the messaging and queuing functions used by your application programs. The amount of processing time is recorded in a file at hourly intervals, and the hourly records are totalled at the end of a month. In this way, the total amount of time that has been used by the WebSphere MQ for z/OS product on your behalf is computed, and used to determine how much you should pay for your use of the WebSphere MQ for z/OS product that month.

MULC is implemented as follows:

- When WebSphere MQ for z/OS is installed, it identifies itself to z/OS , and requests that the *System Management Facilities (SMF)* mechanism within z/OS is to automatically measure how much processor time is used by the WebSphere MQ for z/OS product.
- When enabled, the z/OS usage measurement facility collects usage figures for each hour of the day, and generates usage records that are added to a report file on disk.
- At the end of one full month, these usage records are collected by a program, which generates a report of product usage for the month. This report is used to determine the charge for the WebSphere MQ for z/OS product.

You can find more details on MULC in the *MVS Support for Measured License Charges* manual.

Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing,
IBM Corporation,
North Castle Drive,
Armonk, NY 10504-1785,
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation,
Licensing,
2-31 Roppongi 3-chome, Minato-k,u
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AIX	CICS	DB2
DFSMS	DFSMSdss	IBM
IBMLink	IMS	IMS/ESA
Language Environment	Lotus Notes	MQSeries
MVS	MVS/DFP	NetView
OS/390	RACF	RAMAC
Redbooks	RMF	S/390
SupportPac	VTAM	WebSphere
z/OS	zSeries	

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Index

D

E

F

G

Q

R

S

T

U

V

W

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44-1962-816151
  - From within the U.K., use 01962-816151
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink™: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

GC34-6926-01

Spine information:

WebSphere MQ for z/OS

z/OS Concepts and Planning Guide

Version 7.0