8.0

Configuring IBM MQ

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 631.

This edition applies to version 8 release 0 of IBM® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Configuring

Create one or more queue managers on one or more computers, and configure them on your development, test, and production systems to process messages that contain your business data.

Before you configure IBM MQ, read about the IBM MQ concepts in IBM MQ Technical overview. Read about how to plan your IBM MQ environment in Planning.

There are a number of different methods that you can use to create, configure, and administer your queue managers and their related resources in IBM MQ. These methods include command line interfaces, a graphical user interface, and an administration API. For more information about these interfaces, see Administering IBM MQ .

For instructions on how to create, start, stop, and delete a queue manager, see “Creating and managing queue managers on distributed platforms” on page 5.

For information about how to create the components required to connect your IBM MQ installations and applications together, see “Configuring distributed queuing” on page 122.

For instructions on how to connect your clients to an IBM MQ server by using different methods, see “Configuring connections between the client and server” on page 14.

For instructions on how to configure a queue manager cluster, see “Configuring a queue manager cluster” on page 215.

You can change the behavior of IBM MQ or a queue manager by changing configuration information. For more information, see “Changing IBM MQ and queue manager configuration information” on page 80. In general, you do not need to restart a queue manager for any configuration changes to take effect, except for when stated in this product documentation.

z/OS For instructions on how to configure IBM MQ for z/OS®, see “Configuring queue managers on z/OS” on page 522.

Related concepts

IBM MQ technical overview

“Configuring queue managers on z/OS” on page 522
Use these instructions to configure queue managers on IBM MQ for z/OS.

Related tasks

Administering local IBM MQ objects

Administering remote IBM MQ objects

IBM i Administering IBM i

z/OS Administering IBM MQ for z/OS

Planning

z/OS Planning your IBM MQ environment on z/OS

Creating and managing queue managers on distributed platforms

Before you can use messages and queues, you must create and start at least one queue manager and its associated objects.

Creating a queue manager

A queue manager manages the resources associated with it, in particular the queues that it owns. It provides queuing services to applications for Message queuing Interface (MQI) calls and commands to create, modify, display, and delete IBM MQ objects.

To create a queue manager, you use the IBM MQ control command **crtmqm** (described in **crtmqm**). The **crtmqm** command automatically creates the required default objects and system objects (described in System default objects). Default objects form the basis of any object definitions that you make; system objects are required for queue manager operation. When you have created a queue manager and its objects, use the `strmqm` command to start the queue manager.

Note: IBM MQ does not support machine names that contain spaces. If you install IBM MQ on a computer with a machine name that contains spaces, you cannot create any queue managers.

Before you can create a queue manager, there are several points you must consider (especially in a production environment). Work through the following checklist:

The installation associated with the queue manager

The **crtmqm** command automatically associates a queue manager with the installation from which the **crtmqm** command was issued. For commands that operate on a queue manager, you must issue the command from the installation associated with the queue manager. You can change the associated installation of a queue manager using the setmqm command. Note the Windows installer does not add the user that performs the install to the mqm group, for more details, see Authority to administer IBM MQ on UNIX, Linux® and Windows systems.

Naming conventions

Use uppercase names so that you can communicate with queue managers on all platforms. Remember that names are assigned exactly as you enter them. To avoid the inconvenience of lots of typing, do not use unnecessarily long names.

Specify a unique queue manager name

When you create a queue manager, ensure that no other queue manager has the same name *anywhere* in your network. Queue manager names are not checked when the queue manager is created, and names that are not unique prevent you from creating channels for distributed queuing. Also, if you use the network for publish/subscribe messaging, subscriptions are associated with the queue manager name that created them. Therefore if queue managers in the cluster or hierarchy have the same name, it can result in publications not reaching them.

One way of ensuring uniqueness is to prefix each queue manager name with its own unique node name. For example, if a node is called `ACCOUNTS`, you can name your queue manager `ACCOUNTS.SATURN.QUEUE.MANAGER`, where `SATURN` identifies a particular queue manager and `QUEUE.MANAGER` is an extension you can give to all queue managers. Alternatively, you can omit this, but note that `ACCOUNTS.SATURN` and `ACCOUNTS.SATURN.QUEUE.MANAGER` are *different* queue manager names.

If you are using IBM MQ for communication with other enterprises, you can also include your own enterprise name as a prefix. This is not done in the examples, because it makes them more difficult to follow.

Note: Queue manager names in control commands are case-sensitive. This means that you are allowed to create two queue managers with the names `jupiter.queue.manager` and `JUPITER.queue.manager`. However, it is better to avoid such complications.

Limit the number of queue managers

You can create as many queue managers as resources allow. However, because each queue manager requires its own resources, it is generally better to have one queue manager with 100 queues on a node than to have ten queue managers with ten queues each.

In production systems, many processors can be exploited with a single queue manager, but larger server machines might run more effectively with multiple queue managers.

Specify a default queue manager

Each node should have a default queue manager, though it is possible to configure IBM MQ on a node without one. The default queue manager is the queue manager that applications connect to if they do not specify a queue manager name in an `MQCONN` call. It is also the queue manager that processes MQSC commands when you invoke the `runmqsc` command without specifying a queue manager name.

Specifying a queue manager as the default *replaces* any existing default queue manager specification for the node.

Changing the default queue manage can affect other users or applications. The change has no effect on currently-connected applications, because they can use the handle from their original connect call in any further MQI calls. This handle ensures that the calls are directed to the same queue manager. Any applications connecting *after* you have changed the default queue manager connect to the new default queue manager. This might be what you intend, but you should take this into account before you change the default.

Creating a default queue manager is described in “Creating a default queue manager” on page 8.

Specify a dead-letter queue

The dead-letter queue is a local queue where messages are put if they cannot be routed to their intended destination.

It is important to have a dead-letter queue on each queue manager in your network. If you do not define one, errors in application programs might cause channels to be closed, and replies to administration commands might not be received.

For example, if an application tries to put a message on a queue on another queue manager, but gives the wrong queue name, the channel is stopped and the message remains on the transmission queue. Other applications cannot then use this channel for their messages.

The channels are not affected if the queue managers have dead-letter queues. The undelivered message is put on the dead-letter queue at the receiving end, leaving the channel and its transmission queue available.

When you create a queue manager, use the -u flag to specify the name of the dead-letter queue. You can also use an MQSC command to alter the attributes of a queue manager that you have already defined to specify the dead-letter queue to be used. See Working with queue managers for an example of the MQSC command ALTER.

Specify a default transmission queue

A transmission queue is a local queue on which messages in transit to a remote queue manager are queued before transmission. The default transmission queue is the queue that is used when no transmission queue is explicitly defined. Each queue manager can be assigned a default transmission queue.

When you create a queue manager, use the -d flag to specify the name of the default transmission queue. This does not actually create the queue; you have to do this explicitly later on. See Working with local queues for more information.

Specify the logging parameters you require

You can specify logging parameters on the `crtmqm` command, including the type of logging, and the path and size of the log files.

In a development environment, the default logging parameters should be adequate. However, you can change the defaults if, for example:

- You have a low-end system configuration that cannot support large logs.
- You anticipate a large number of long messages being on your queues at the same time.
- You anticipate a lot of persistent messages passing through the queue manager.

Once you have set the logging parameters, some of them can only be changed by deleting the queue manager and recreating it with the same name but with different logging parameters.

For more information about logging parameters, see “Availability, recovery and restart” on page 367.

UNIX **For IBM MQ for UNIX systems only**

You can create the queue manager directory `/var/mqm/qmgrs/<qmgr>`, even on a separate local file system, before you use the `crtmqm` command. When you use `crtmqm` , if the `/var/mqm/qmgrs/<qmgr>` directory exists, is empty, and is owned by mqm, it is used for the queue manager data. If the directory is not owned by mqm, the creation fails with a First Failure Support Technology (FFST) message. If the directory is not empty, a new directory is created.

Related concepts

"Configuring queue managers on z/OS" on page 522
Use these instructions to configure queue managers on IBM MQ for z/OS.

"Configuring" on page 5
Create one or more queue managers on one or more computers, and configure them on your development, test, and production systems to process messages that contain your business data.

"Backing up configuration files after creating a queue manager" on page 10
IBM MQ configuration information is stored in configuration files on Windows, UNIX and Linux systems.

"Starting a queue manager" on page 11
When you create a queue manager, you must start it to enable it to process commands or MQI calls.

"Stopping a queue manager" on page 11
There are three ways to stop a queue manager: a quiesced shutdown, and immediate shutdown, and a preemptive shutdown.

"Restarting a queue manager" on page 13
You can use the **strmqm** command to restart a queue manager, or on IBM MQ for Windows and IBM MQ for Linux x86-64 systems, restart a queue manager from IBM MQ Explorer.

"Changing IBM MQ and queue manager configuration information" on page 80
Change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

Related tasks

Creating a queue manager called QM1

"Making an existing queue manager the default" on page 9
You can make an existing queue manager the default queue manager. The way you do this depends on the platform you are using.

"Deleting a queue manager" on page 13
You can delete a queue manager using the **dltmqm** command or by using the IBM MQ Explorer.

Related reference

System and default objects

distributed Creating a default queue manager

The default queue manager is the queue manager that applications connect to if they do not specify a queue manager name in an MQCONN call. It is also the queue manager that processes MQSC commands when you invoke the **runmqsc** command without specifying a queue manager name. To create a queue manager, you use the IBM MQ control command **crtmqm**.

Before you begin

Before creating a default queue manager, read through the considerations described in "Creating and managing queue managers on distributed platforms" on page 5.

UNIX When you use **crtmqm** to create a queue manager on UNIX, if the `/var/mqm/qmgrs/<qmgr>` directory already exists, is owned by mqm, and is empty, it is used for the queue manager data. If the directory is not owned by mqm, the creation of the queue manager fails with a First Failure Support Technology (FFST) message. If the directory is not empty, a new directory is created for the queue manager data.

This consideration applies even when the `/var/mqm/qmgrs/<qmgr>` directory already exists on a separate local file system.

About this task

When you create a queue manager by using the **crtmqm** command, the command automatically creates the required default objects and system objects. Default objects form the basis of any object definitions that you make and system objects are required for queue manager operation.

By including the relevant parameters in the command, you can also define, for example, the name of the default transmission queue to be used by the queue manager, and the name of the dead letter queue.

Windows On Windows, you can use the **sax** option of the **crtmqm** command to start multiple instances of the queue manager.

For more information about the **crtmqm** command and its syntax, see **crtmqm**.

Procedure

- To create a default queue manager, use the **crtmqm** command with the **-q** flag.

  The following example of the **crtmqm** command creates a default queue manager called SATURN.QUEUE.MANAGER:

```
crtmqm -q -d MY.DEFAULT.XMIT.QUEUE -u SYSTEM.DEAD.LETTER.QUEUE SATURN.QUEUE.MANAGER
```

  where:

  -q
  Indicates that this queue manager is the default queue manager.

  -d MY.DEFAULT.XMIT.QUEUE
  Is the name of the default transmission queue to be used by this queue manager.

  Note: IBM MQ does not create a default transmission queue for you; you have to define it yourself.

  -u SYSTEM.DEAD.LETTER.QUEUE
  Is the name of the default dead-letter queue created by IBM MQ on installation.

  SATURN.QUEUE.MANAGER
  Is the name of this queue manager. This must be the last parameter specified on the `crtmqm` command.

What to do next

When you have created a queue manager and its objects, use the **strmqm** command to start the queue manager.

Related concepts

"Backing up configuration files after creating a queue manager" on page 10
IBM MQ configuration information is stored in configuration files on Windows, UNIX and Linux systems.

Working with queue managers
Working with local queues

Related reference

System and default objects

Making an existing queue manager the default

You can make an existing queue manager the default queue manager. The way you do this depends on the platform you are using.

IBM MQ for Windows and IBM MQ for Linux (x86 and x86-64 platforms) systems

About this task

Use the following instructions to make an existing queue manager the default queue manager on IBM MQ for Windows and IBM MQ for Linux (x86 and x86-64 platforms) systems:

Procedure

1. Open the IBM MQ Explorer.
2. Right-click `IBM WebSphere MQ`, then select `Properties...`. The Properties for IBM MQ panel is displayed.
3. Type the name of the default queue manager into the Default queue manager name field.
4. Click OK.

UNIX and Linux systems

About this task

When you create a default queue manager, its name is inserted in the `Name` attribute of the `DefaultQueueManager` stanza in the IBM MQ configuration file (mqs.ini). The stanza and its contents are automatically created if they do not exist.

Procedure

- To make an existing queue manager the default, change the queue manager name on the `Name` attribute to the name of the new default queue manager. You can do this manually, using a text editor.
- If you do not have a default queue manager on the node, and you want to make an existing queue manager the default, create the *`DefaultQueueManager`* stanza with the required name yourself.
- If you accidentally make another queue manager the default and want to revert to the original default queue manager, edit the `DefaultQueueManager` stanza in mqs.ini, replacing the unwanted default queue manager with that of the one you want.

What to do next

See "Changing IBM MQ and queue manager configuration information" on page 80 for information about configuration files.

Backing up configuration files after creating a queue manager

IBM MQ configuration information is stored in configuration files on Windows, UNIX and Linux systems.

On Windows and Linux (x86 and x86-64) systems use MQ Explorer to make changes to the configuration files.

On Windows systems you can also use the `amqmdain` command to make changes to the configuration files. See, amqmdain

There are two types of configuration file:

- When you install the product, the IBM MQ configuration file (mqs.ini) is created. It contains a list of queue managers that is updated each time you create or delete a queue manager. There is one mqs.ini file per node.
- When you create a new queue manager, a new queue manager configuration file (qm.ini) is automatically created. This contains configuration parameters for the queue manager.

V 8.0.0.4 If you have installed the AMQP service, then there is an additional configuration file that you must back up:

- `amqp_win.properties` (Windows)
- `amqp_unix.properties` (UNIX/Linux)

After creating a queue manager, back up your configuration files. Then, if you create another queue manager that causes you problems, you can reinstate the backups when you have removed the source of the problem. As a general rule, back up your configuration files each time you create a new queue manager.

For more information about configuration files, see "Changing IBM MQ and queue manager configuration information" on page 80.

Starting a queue manager

When you create a queue manager, you must start it to enable it to process commands or MQI calls.

To start a queue manager, use the **strmqm** command.

Note: You must use the **strmqm** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmq -o installation` command.

For example, to start a queue manager QMB enter the following command:

```
strmqm QMB
```

On IBM MQ for Windows and IBM MQ for Linux (x86 and x86-64 platforms) systems, you can start a queue manager as follows:

1. Open the IBM MQ Explorer.
2. Select the queue manager from the Navigator View.
3. Click `Start` . The queue manager starts.

If the queue manager start-up takes more than a few seconds IBM MQ issues information messages intermittently detailing the start-up progress.

The `strmqm` command does not return control until the queue manager has started and is ready to accept connection requests.

Starting a queue manager automatically

In IBM MQ for Windows you can start a queue manager automatically when the system starts using the IBM MQ Explorer. For more information, see Administration using the MQ Explorer .

Stopping a queue manager

There are three ways to stop a queue manager: a quiesced shutdown, and immediate shutdown, and a preemptive shutdown.

Use the **endmqm** command to stop a queue manager.

Note: You must use the **endmqm** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmq -o installation` command.

For example, to stop a queue manager called QMB, enter the following command:

```
endmqm QMB
```

On IBM MQ for Windows and IBM MQ for Linux (x86 and x86-64 platforms) systems, you can stop a queue manager as follows:

1. Open the IBM MQ Explorer.
2. Select the queue manager from the Navigator View.
3. Click `Stop...`. The End Queue Manager panel is displayed.
4. Select Controlled, or Immediate.
5. Click `OK`. The queue manager stops.

Quiesced shutdown

By default, the **endmqm** command performs a quiesced shutdown of the specified queue manager. This might take a while to complete. A quiesced shutdown waits until all connected applications have disconnected.

Use this type of shutdown to notify applications to stop. If you issue:

```
endmqm -c QMB
```

you are not told when all applications have stopped. (An `endmqm -c QMB` command is equivalent to an `endmqm QMB` command.)

However, if you issue:

```
endmqm -w QMB
```

the command waits until all applications have stopped and the queue manager has ended.

Immediate shutdown

For an immediate shutdown any current MQI calls are allowed to complete, but any new calls fail. This type of shutdown does not wait for applications to disconnect from the queue manager.

For an immediate shutdown, type:

```
endmqm -i QMB
```

Preemptive shutdown

Note: Do not use this method unless all other attempts to stop the queue manager using the **endmqm** command have failed. This method can have unpredictable consequences for connected applications.

If an immediate shutdown does not work, you must resort to a *preemptive* shutdown, specifying the `-p` flag. For example:

```
endmqm -p QMB
```

This stops the queue manager immediately. If this method still does not work, see Stopping a queue manager manually for an alternative solution.

For a detailed description of the **endmqm** command and its options, see endmqm.

If you have problems shutting down a queue manager

Problems in shutting down a queue manager are often caused by applications. For example, when applications:

- Do not check MQI return codes properly

- Do not request notification of a quiesce
- Terminate without disconnecting from the queue manager (by issuing an `MQDISC` call)

If a problem occurs when you stop the queue manager, you can break out of the **endmqm** command using Ctrl-C. You can then issue another **endmqm** command, but this time with a flag that specifies the type of shutdown that you require.

Restarting a queue manager

You can use the **strmqm** command to restart a queue manager, or on IBM MQ for Windows and IBM MQ for Linux x86-64 systems, restart a queue manager from IBM MQ Explorer.

To restart a queue manager, type:

```
strmqm saturn.queue.manager
```

On IBM MQ for Windows and IBM MQ for Linux x86-64 systems, you can restart a queue manager in the same way as starting it, as follows:

1. Open the IBM MQ Explorer.
2. Select the queue manager from the Navigator View.
3. Click `Start` . The queue manager restarts.

If the queue manager restart takes more than a few seconds IBM MQ issues information messages intermittently detailing the start-up progress.

Deleting a queue manager

You can delete a queue manager using the **dltmqm** command or by using the IBM MQ Explorer.

Before you begin

Stop the queue manager.

Procedure

- Issue the following command: `dltmqm QMB`

  Note: You must use the **dltmqm** command from the installation associated with the queue manager that you are working with. You can find out which installation a queue manager is associated with using the `dspmq -o installation` command.

Steps for deleting a queue manager

About this task

On IBM MQ for Windows and IBM MQ for Linux (x86 and x86-64 platforms) systems, you can delete a queue manager as follows:

Procedure

1. Open the IBM MQ Explorer.
2. In the Navigator view, select the queue manager.
3. If the queue manager is not stopped, stop it.
   a) Right-click the queue manager.
   b) Click **Stop**.
4. Right-click the queue manager.
5. Click **Delete**.

Results

The queue manager is deleted.

Attention:

- Deleting a queue manager is a drastic step, because you also delete all resources associated with the queue manager, including all queues and their messages and all object definitions. If you use the **dltmqm** command, there is no displayed prompt that allows you to change your mind; when you press the Enter key all the associated resources are lost.
- In IBM MQ for Windows, deleting a queue manager also removes the queue manager from the automatic startup list (described in "Starting a queue manager" on page 11). When the command has completed, a `IBM MQ queue manager ending` message is displayed; you are not told that the queue manager has been deleted.
- Deleting a cluster queue manager does not remove it from the cluster. See the note in the description of **dltmqm** for more information.

For a description of the **dltmqm** command and its options, see dltmqm. Ensure that only trusted administrators have the authority to use this command. (For information about security, see Setting up security on Windows, UNIX and Linux systems.)

Configuring connections between the client and server

To configure the communication links between IBM MQ MQI clients and servers, decide on your communication protocol, define the connections at both ends of the link, start a listener, and define channels.

In IBM MQ, the logical communication links between objects are called *channels*. The channels used to connect IBM MQ MQI clients to servers are called MQI channels. You set up channel definitions at each end of your link so that your IBM MQ application on the IBM MQ MQI client can communicate with the queue manager on the server. For a detailed description of how to do this, see User defined channels.

Before you define your MQI channels, you must:

1. Decide on the form of communication that you are going to use. See "Which communication type to use" on page 14.
2. Define the connection at each end of the channel:

   To define the connection, you must:

   - Configure the connection.
   - Record the values of the parameters that you need for the channel definitions.
   - Enable the server to detect incoming network requests from your IBM MQ MQI client, by starting a *listener*.

Which communication type to use

Different platforms support different transmission protocols. Your choice of transmission protocol depends on your combination of IBM MQ MQI client and server platforms.

There are up to four types of transmission protocol for MQI channels depending on your client and server platforms:

- LU 6.2
- NetBIOS
- SPX
- TCP/IP

When you define your MQI channels, each channel definition must specify a transmission protocol (transport type) attribute. A server is not restricted to one protocol, so different channel definitions can specify different protocols. For IBM MQ MQI clients, it might be useful to have alternative MQI channels using different transmission protocols.

Your choice of transmission protocol might be restricted by your particular combination of IBM MQ MQI client and server platforms. The possible combinations are shown in the following table.

Table 1. Transmission protocols - combination of IBM MQ MQI client and server platforms

Transmission protocol	IBM MQ MQI client	IBM MQ server
TCP/IP	IBM i IBM i UNIX systems Windows	IBM i IBM i UNIX systems Windows z/OS
LU 6.2	UNIX systems [1] Windows	IBM i IBM i UNIX systems [1] Windows z/OS z/OS
NetBIOS	Windows	Windows
SPX	Windows	Windows

Note:

1. Except Linux for Power Systems

For more about setting up different types of connections, see the following links:

- "TCP/IP connection limits" on page 18
- "Defining a TCP connection on Windows" on page 183
- "Defining a TCP connection on UNIX and Linux" on page 190
- IBM i "Defining a TCP connection on IBM i" on page 209
- z/OS "Defining a TCP connection on z/OS" on page 608
- "Defining an LU 6.2 connection on Windows" on page 185
- "Defining an LU 6.2 connection on UNIX and Linux" on page 193
- IBM i "Defining an LU 6.2 connection on IBM i" on page 210
- z/OS "Defining an LU6.2 connection for z/OS using APPC/MVS" on page 610
- "Defining a NetBIOS connection on Windows" on page 187

Related concepts

"Configuring an extended transactional client" on page 18
This collection of topics describes how to configure the extended transactional function for each category of transaction manager.

"Defining MQI channels" on page 28
To create a new channel, you have to create **two** channel definitions, one for each end of the connection, using the same channel name and compatible channel types. In this case, the channel types are *server-connection* and *client-connection*.

"Creating server-connection and client-connection definitions on different platforms" on page 34

You can create each channel definition on the computer to which it applies. There are restrictions on how you can create channel definitions on a client computer.

“Creating server-connection and client-connection definitions on the server” on page 37
You can create both definitions on the server, then make the client-connection definition available to the client.

“Channel-exit programs for MQI channels” on page 43
Three types of channel exit are available to the IBM MQ MQI client environment on UNIX, Linux and Windows systems.

“Connecting a client to a queue-sharing group” on page 47
You can connect a client to a queue-sharing group by creating an MQI channel between a client and a queue manager on a server that is a member of a queue-sharing group.

“Configuring a client using a configuration file” on page 47
Configure your clients by using attributes in a text file. These attributes can be overridden by environment variables or in other platform-specific ways.

Related tasks

Connecting IBM MQ MQI client applications to queue managers

Related reference

DISPLAY CHLAUTH

SET CHLAUTH

Which communication type to use

Different platforms support different communication protocols. Your choice of transmission protocol depends on your combination of IBM MQ MQI client and server platforms.

There are four types of communication for MQI channels on different platforms:

- LU 6.2
- NetBIOS
- SPX
- TCP/IP

When you define your MQI channels, each channel definition must specify a transmission protocol (transport type) attribute. A server is not restricted to one protocol, so different channel definitions can specify different protocols. For IBM MQ MQI clients, it might be useful to have alternative MQI channels using different transmission protocols.

Your choice of transmission protocol also depends on your particular combination of IBM MQ client and server platforms. The possible combinations are shown in the following table.

Note:

1. Except Linux (POWER platform)

Table 2. Transmission protocols - combination of IBM MQ client and server platforms

Transmission protocol	IBM MQ MQI client	IBM MQ server
TCP/IP	IBM i IBM i UNIX systems Windows	IBM i IBM i UNIX systems Windows z/OS z/OS

Transmission protocol	IBM MQ MQI client	IBM MQ server
LU 6.2	UNIX systems [1] Windows	IBM i IBM i UNIX systems [1] Windows z/OS z/OS
NetBIOS	Windows	Windows
SPX	Windows	Windows

Table 2. Transmission protocols - combination of IBM MQ client and server platforms (continued)

Related concepts

"Defining a TCP connection on Windows" on page 183
Define a TCP connection by configuring a channel at the sending end to specify the address of the target, and by running a listener program at the receiving end.

"Defining a TCP connection on UNIX and Linux" on page 190
The channel definition at the sending end specifies the address of the target. The listener or inet daemon is configured for the connection at the receiving end.

"Defining a TCP connection on IBM i" on page 209
You can define a TCP connection within the channel definition using the Connection Name field.

"Defining a TCP connection on z/OS" on page 608
To define a TCP connection, there are a number of settings to configure.

"Defining an LU 6.2 connection on Windows" on page 185
SNA must be configured so that an LU 6.2 conversation can be established between the two machines.

"Defining an LU 6.2 connection on UNIX and Linux" on page 193
SNA must be configured so that an LU 6.2 conversation can be established between the two machines.

"Defining an LU 6.2 connection on IBM i" on page 210
Define the LU 6.2 communications details by using a mode name, TP name, and connection name of a fully qualified LU 6.2 connection.

"Defining a NetBIOS connection on Windows" on page 187
IBM MQ uses three types of NetBIOS resource when establishing a NetBIOS connection to another IBM MQ product: sessions, commands, and names. Each of these resources has a limit, which is established either by default or by choice during the installation of NetBIOS.

Related reference

"TCP/IP connection limits" on page 18
The number of outstanding connection requests that can be queued at a single TCP/IP port depends on the platform. An error occurs if the limit is reached.

"Defining an LU6.2 connection for z/OS using APPC/MVS" on page 610
To define an LU6.2 connection there are a number of settings to configure.

Defining a TCP/IP connection

Specifying a transport type of TCP on the channel definition at the client end. Start a listener program on the server.

Specify a TCP/IP connection at the client by specifying a transport type of TCP on the channel definition.

Receiving channel programs are started in response to a startup request from the sending channel. To do this, a listener program has to be started to detect incoming network requests and start the associated channel. The procedure for starting a listener program depends on the server platform.

See the related topics for your client and server platforms.

TCP/IP connection limits

The number of outstanding connection requests that can be queued at a single TCP/IP port depends on the platform. An error occurs if the limit is reached.

This connection limit is not the same as the maximum number of clients you can attach to an IBM MQ server. You can connect more clients to a server, up to the level determined by the server system resources. The backlog values for connection requests are shown in the following table:

Table 3. Maximum outstanding connection requests queued at a TCP/IP port

Server platform	**Maximum connection requests**
AIX®	100
HP-UX	20
Linux	100
IBM	255
Solaris	100
Windows Server	100
Windows Workstation	100
z/OS	255

If the connection limit is reached, the client receives a return code of MQRC_HOST_NOT_AVAILABLE from the `MQCONN` call, and an AMQ9202 error in the client error log (`/var/mqm/errors/AMQERR0n.LOG` on UNIX and Linux systems or `amqerr0n.log` in the errors subdirectory of the IBM MQ client installation on Windows). If the client retries the `MQCONN` request, it might be successful.

To increase the number of connection requests you can make, and avoid error messages being generated by this limitation, you can have multiple listeners each listening on a different port, or have more than one queue manager.

Defining a NetBIOS connection

NetBIOS connections apply only to Windows systems.

A NetBIOS connection applies only to a client and server running Windows. See Defining a NetBIOS connection.

Configuring an extended transactional client

This collection of topics describes how to configure the extended transactional function for each category of transaction manager.

For each platform, the extended transactional client provides support for the following external transaction managers:

XA-compliant transaction managers
: The extended transactional client provides the XA resource manager interface to support XA-compliant transaction managers such as CICS® and Tuxedo.

Microsoft Transaction Server (Windows systems only)
: On Windows systems only, the XA resource manager interface also supports Microsoft Transaction Server (MTS). The IBM MQ MTS support supplied with the extended transactional client provides the bridge between MTS and the XA resource manager interface.

WebSphere® Application Server

Earlier versions of IBM MQ supported WebSphere Application Server Version 4 or Version 5, and required you to carry out certain configuration tasks to use the extended transactional client. WebSphere Application Server Version 6 and later includes an IBM MQ messaging provider, so you do not need to use the extended transactional client.

Related concepts

"Configuring XA-compliant transaction managers" on page 19
First configure the IBM MQ base client, then configure the extended transactional function using the information in these topics.

"Microsoft Transaction Server" on page 27
No additional configuration is required before you can use MTS as a transaction manager. However, there are some points to note.

Configuring XA-compliant transaction managers

First configure the IBM MQ base client, then configure the extended transactional function using the information in these topics.

Note: This section assumes that you have a basic understanding of the XA interface as published by The Open Group in *Distributed Transaction Processing: The XA Specification*.

To configure an extended transactional client, you must first configure the IBM MQ base client as described in Installing an IBM MQ client. Using the information in this section, you can then configure the extended transactional function for an XA-compliant transaction manager such as CICS and Tuxedo.

A transaction manager communicates with a queue manager as a resource manager using the same MQI channel as that used by the client application that is connected to the queue manager. When the transaction manager issues a resource manager (xa_) function call, the MQI channel is used to forward the call to the queue manager, and to receive the output back from the queue manager.

Either the transaction manager can start the MQI channel by issuing an xa_open call to open the queue manager as a resource manager, or the client application can start the MQI channel by issuing an MQCONN or MQCONNX call.

- If the transaction manager starts the MQI channel, and the client application later calls MQCONN or MQCONNX on the same thread, the MQCONN or MQCONNX call completes successfully and a connection handle is returned to the application. The application does not receive a MQCC_WARNING completion code with an MQRC_ALREADY_CONNECTED reason code.
- If the client application starts the MQI channel, and the transaction manager later calls xa_open on the same thread, the xa_open call is forwarded to the queue manager using the MQI channel.

In a recovery situation following a failure, when no client applications are running, the transaction manager can use a dedicated MQI channel to recover any incomplete units of work in which the queue manager was participating at the time of the failure.

Note the following conditions when using an extended transactional client with an XA-compliant transaction manager:

- Within a single thread, a client application can be connected to only one queue manager at a time. This restriction applies only when using an extended transactional client; a client application that is using an IBM MQ base client can be connected to more than one queue manager concurrently within a single thread.
- Each thread of a client application can connect to a different queue manager.
- A client application cannot use shared connection handles.

To configure the extended transactional function, you must provide the following information to the transaction manager for each queue manager that acts as a resource manager:

- An xa_open string
- A pointer to an XA switch structure

When the transaction manager calls xa_open to open the queue manager as a resource manager, it passes the xa_open string to the extended transactional client as the argument, *xa_info*, on the call. The extended transactional client uses the information in the xa_open string in the following ways:

- To start an MQI channel to the server queue manager, if the client application has not already started one
- To check that the queue manager that the transaction manager opens as a resource manager is the same as the queue manager to which the client application connects
- To locate the transaction manager's ax_reg and ax_unreg functions, if the queue manager uses dynamic registration

For the format of an xa_open string, and for more details about how the information in the xa_open string is used by an extended transactional client, see "The format of an xa_open string" on page 21.

An XA switch structure enables the transaction manager to locate the xa_ functions provided by the extended transactional client, and specifies whether the queue manager uses dynamic registration. For information about the XA switch structures supplied with an extended transactional client, see "The XA switch structures" on page 25.

For information about how to configure the extended transactional function for a particular transaction manager, and for any other information about using the transaction manager with an extended transactional client, see the following sections:

- "Configuring an extended transactional client for CICS" on page 26
- "Configuring an extended transactional client for Tuxedo" on page 27

Related concepts

"The CHANNEL, TRPTYPE, CONNAME, and QMNAME parameters of the xa_open string" on page 23
Use this information to understand how the extended transactional client uses these parameters to determine the queue manager to connect to.

"Additional error processing for xa_open" on page 24
The xa_open call fails in certain circumstances.

Related tasks

"Using the extended transactional client with SSL channels" on page 25
You cannot set up an SSL channel using the xa_open string. Follow these instructions to use the client channel definition table (ccdt).

Related reference

"The TPM and AXLIB parameters" on page 24
An extended transactional client uses the TPM and AXLIB parameters to locate the transaction manager's ax_reg and ax_unreg functions. These functions are used only if the queue manager uses dynamic registration.

"Recovery following a failure in extended transactional processing" on page 24
Following a failure, a transaction manager must be able to recover any incomplete units of work. To do this, the transaction manager must be able to open as a resource manager any queue manager that was participating in an incomplete unit of work at the time of the failure.

IBM WebSphere MQ for z/OS considerations for extended transactional client connections

Some XA transaction managers use sequences of transaction coordination calls which are incompatible with the features normally available to clients connecting to IBM WebSphere MQ for z/OS.

Where an incompatible sequence is detected, IBM WebSphere MQ for z/OS might issue an abend for the connection and return an error response to the client.

For example, `xa_prepare` receives abend 5C6-00D4007D, with return code -3 (XAER_RMERR) returned to the client.

For transaction managers which encounter this situation, take the following actions to allow the transaction manager to interact with IBM WebSphere MQ for z/OS:

- Apply the fix for APAR PI49236.
- Enable the change provided by PI49236 for the server-connection channel used by the transaction manager.

  You enable the change by specifying the keyword CSQSERVICE1 (in upper case) anywhere in the description field of the SVRCONN channel.

Note that channels with the CSQSERVICE1 keyword have the following restrictions:

- GROUP unit of recovery disposition is not permitted. Only QMGR unit of recovery disposition is allowed.

  An xa_open call specifying the queue sharing group name in the **xa_info** parameter fails with *xaer_inval*.
- The *MQGMO_LOCK* and *MQGMO_UNLOCK* options are not permitted. An MQGET call with *MQGMO_LOCK* or *MQGMO_UNLOCK* fails with MQRC_ENVIRONMENT_ERROR.

Related concepts

"Configuring XA-compliant transaction managers" on page 19
First configure the IBM MQ base client, then configure the extended transactional function using the information in these topics.

The format of an xa_open string

An xa_open string contains pairs of defined parameter names and values.

An xa_open string has the following format:

```
parm_name1 = parm_value1, parm_name2 = parm_value2, ...
```

where *parm_name* is the name of a parameter and *parm_value* is the value of a parameter. The names of the parameters are not case-sensitive but, unless stated otherwise, the values of the parameters are case-sensitive. You can specify the parameters in any order.

The names, meanings, and valid values of the parameters are as follows:

Name
: **Meaning and valid values**

CHANNEL
: The name of an MQI channel.

  This is an optional parameter. If this parameter is supplied, the CONNAME parameter must also be supplied.

TRPTYPE
: The communications protocol for the MQI channel. The following are valid values:

  LU62
  : SNA LU 6.2

  NETBIOS
  : NetBIOS

  SPX
  : IPX/SPX

  TCP
  : TCP/IP

  This is an optional parameter. If it is omitted, the default value of TCP is assumed. The values of the parameter are not case-sensitive.

CONNAME
: The network address of the queue manager at the server end of the MQI channel. The valid values of this parameter depend on the value of the TRPTYPE parameter:

LU62
A symbolic destination name, which identifies a CPI-C side information entry.

The network qualified name of a partner LU is not a valid value, nor is a partner LU alias. This is because there are no additional parameters to specify a transaction program (TP) name and a mode name.

NETBIOS
A NetBIOS name.

SPX
A 4 byte network address, a 6 byte node address, and an optional 2 byte socket number. These values must be specified in hexadecimal notation. A period must separate the network and node addresses, and the socket number, if supplied, must be enclosed in parentheses. For example:

```
0a0b0c0d.804abcde23a1(5e86)
```

If the socket number is omitted, the default value of 5e86 is assumed.

TCP
A host name or an IP address, optionally followed by a port number in parentheses. If the port number is omitted, the default value of 1414 is assumed.

This is an optional parameter. If this parameter is supplied, the CHANNEL parameter must also be supplied.

QMNAME
The name of the queue manager at the server end of the MQI channel. The name cannot be blank or a single asterisk (*), nor can the name start with an asterisk. This means that the parameter must identify a specific queue manager by name.

This is a mandatory parameter.

When a client application is connected to a specific queue manager any transaction recovery must be processed by the same queue manager.

If the application is connecting to a z/OS queue manager then the application can specify either the name of a specific queue manager or the name of a queue-sharing group (QSG). By using the queue manager name or QSG name, the application controls whether it partakes in a transaction with a QMGR unit of recovery disposition or a GROUP unit of recovery disposition. The GROUP unit of recovery disposition enables the recovery of the transaction to be processed on any member of the QSG. To use GROUP units of recovery the **GROUPUR** queue manager attribute must be enabled.

z/OS For further information about using GROUP unit of recovery, see Unit of recovery disposition in a queue-sharing group.

TPM
The transaction manager being used. The valid values are CICS and TUXEDO.

An extended transactional client uses this parameter and the AXLIB parameter for the same purpose. For more information these parameters, see The TPM and AXLIB parameters.

This is an optional parameter. The values of the parameter are not case-sensitive.

AXLIB
The name of the library that contains the transaction manager's ax_reg and ax_unreg functions.

This is an optional parameter.

UID
The user ID that is provided to the queue manager for authentication. If this parameter is supplied, the **PWD** parameter must also be supplied. If the user ID and password supplied are authenticated, the user ID is used for identification of ths transaction manager's connection. The user ID and password populate the MQCSP object on the MQCONNX call.

The **UID** and **PWD** parameters are valid for both client and server bindings.

PWD
: The password that is provided to the queue manager for authentication. If this parameter is supplied, the **UID** parameter must also be supplied.

Warning: In some cases, the password in an MQCSP structure for a client application will be sent across a network in plain text. To ensure that client application passwords are protected appropriately, see IBM MQCSP password protection.

Here is an example of an xa_open string:

```
channel=MARS.SVR,trptype=tcp,conname=MARS(1415),qmname=MARS,tpm=cics
```

The CHANNEL, TRPTYPE, CONNAME, and QMNAME parameters of the xa_open string

Use this information to understand how the extended transactional client uses these parameters to determine the queue manager to connect to.

If the CHANNEL and CONNAME parameters are supplied in the xa_open string, the extended transactional client uses these parameters and the TRPTYPE parameter to start an MQI channel to the server queue manager.

If the CHANNEL and CONNAME parameters are not supplied in the xa_open string, the extended transactional client uses the value of the MQSERVER environment variable to start an MQI channel. If the MQSERVER environment variable is not defined, the extended transactional client uses the entry in the client channel definition identified by the QMNAME parameter.

In each of these cases, the extended transactional client checks that the value of the QMNAME parameter is the name of the queue manager at the server end of the MQI channel. If it is not, the xa_open call fails and the transaction manager reports the failure to the application.

If the application connects to a queue manager at an earlier version than V7.0.1, the xa_open call succeeds but the transaction has a QMGR unit of recovery disposition. z/OS Ensure that applications that require the GROUP unit of recovery disposition connect only to queue managers at V7.0.1 or later.

z/OS If the application uses a QSG name in QMNAME parameter field and the GROUPUR property is disabled on the queue manager to which it connects then the xa_open call fails.

z/OS If the application client is connecting to a z/OS queue manager at V7.0.1 or later it can specify a queue-sharing group (QSG) name for the QMNAME parameter. This allows the application client to participate in a transaction with a GROUP unit of recovery disposition. For more information about the GROUP unit of recovery disposition, see Unit of recovery disposition.

When the client application later calls MQCONN or MQCONNX on the same thread that the transaction manager used to issue the xa_open call, the application receives a connection handle for the MQI channel that was started by the xa_open call. A second MQI channel is not started. The extended transactional client checks that the value of the *QMgrName* parameter on the MQCONN or MQCONNX call is the name of the queue manager at the server end of the MQI channel. If it is not, the MQCONN or MQCONNX call fails with a reason code of MQRC_ANOTHER_Q_MGR_CONNECTED. If the value of the *QMgrName* parameter is blank or a single asterisk (*), or starts with an asterisk, the MQCONN or MQCONNX call fails with a reason code of MQRC_Q_MGR_NAME_ERROR.

If the client application has already started an MQI channel by calling MQCONN or MQCONNX before the transaction manager calls xa_open on the same thread, the transaction manager uses this MQI channel instead. A second MQI channel is not started. The extended transactional client checks that the value of the QMNAME parameter in the xa_open string is the name of the server queue manager. If it is not, the xa_open call fails.

If a client application starts an MQI channel first, the value of the *QMgrName* parameter on the MQCONN or MQCONNX call can be blank or a single asterisk (*), or it can start with an asterisk. Under these circumstances, however, you must ensure that the queue manager to which the application connects is the same as the queue manager that the transaction manager intends to open as a resource manager when it later calls xa_open on the same thread. You might encounter fewer problems, therefore, if the value of the *QMgrName* parameter identifies the queue manager explicitly by name.

The TPM and AXLIB parameters

An extended transactional client uses the TPM and AXLIB parameters to locate the transaction manager's ax_reg and ax_unreg functions. These functions are used only if the queue manager uses dynamic registration.

If the TPM parameter is supplied in an xa_open string, but the AXLIB parameter is not supplied, the extended transactional client assumes a value for the AXLIB parameter based on the value of the TPM parameter. See Table 4 on page 24 for the assumed values of the AXLIB parameter.

Table 4. Assumed values of the AXLIB parameter

Value of TPM	**Platform**	**Assumed value of AXLIB**
CICS	AIX	/usr/lpp/encina/lib/libEncServer.a(EncServer_shr.o)
CICS	HP-UX	/opt/encina/lib/libEncServer.sl
CICS	Solaris	/opt/encina/lib/libEncServer.so
CICS	Windows systems	libEncServer
Tuxedo	AIX	/usr/lpp/tuxedo/lib/libtux.a(libtux.so.60)
Tuxedo	HP-UX	/opt/tuxedo/lib/libtux.sl
Tuxedo	Solaris	/opt/tuxedo/lib/libtux.so.60
Tuxedo	Windows systems	libtux

If the AXLIB parameter is supplied in an xa_open string, the extended transactional client uses its value to override any assumed value based on the value of the TPM parameter. The AXLIB parameter can also be used for a transaction manager for which the TPM parameter does not have a specified value.

Additional error processing for xa_open

The xa_open call fails in certain circumstances.

Topics in this section describe situations in which the xa_open call fails. It also fails if any of the following situations occur:

- There are errors in the xa_open string.
- There is insufficient information to start an MQI channel.
- There is a problem while trying to start an MQI channel (the server queue manager is not running, for example).

Recovery following a failure in extended transactional processing

Following a failure, a transaction manager must be able to recover any incomplete units of work. To do this, the transaction manager must be able to open as a resource manager any queue manager that was participating in an incomplete unit of work at the time of the failure.

Therefore, you must ensure that all incomplete units of work have been resolved before making changes to any configuration information.

Alternatively, you must ensure that the configuration changes do not affect the ability of the transaction manager to open the queue managers it needs to open. The following are examples of such configuration changes:

- Changing the contents of an xa_open string
- Changing the value of the MQSERVER environment variable
- Changing entries in the client channel definition table (CCDT)
- Deleting a server connection channel definition

The XA switch structures

Two XA switch structures are supplied with the extended transactional client on each platform.

These switch structures are:

MQRMIXASwitch
: This switch structure is used by a transaction manager when a queue manager, acting as a resource manager, is not using dynamic registration.

MQRMIXASwitchDynamic
: This switch structure is used by a transaction manager when a queue manager, acting as a resource manager, uses dynamic registration.

These switch structures are located in the libraries shown in Table 5 on page 25.

Table 5. IBM MQ libraries containing the XA switch structures

Platform	Library containing the XA switch structures
AIX HP-UX Linux Solaris	*MQ_INSTALLATION_PATH*/lib/libmqcxa
Windows systems	*MQ_INSTALLATION_PATH*\bin\mqcxa.dll [1]

MQ_INSTALLATION_PATH represents the high-level directory in which IBM MQ is installed.

The name of the IBM MQ resource manager in each switch structure is MQSeries_XA_RMI, but many queue managers can share the same switch structure.

Related concepts

"Dynamic registration and extended transactional processing" on page 25
Using dynamic registration is a form of optimization because it can reduce the number of xa_ function calls issued by the transaction manager.

Dynamic registration and extended transactional processing
Using dynamic registration is a form of optimization because it can reduce the number of xa_ function calls issued by the transaction manager.

If a queue manager does not use dynamic registration, a transaction manager involves the queue manager in every unit of work. The transaction manager does this by calling xa_start, xa_end, and xa_prepare, even if the queue manager has no resources that are updated within the unit of work.

If a queue manager uses dynamic registration, a transaction manager starts by assuming that the queue manager is not involved in a unit of work, and does not call xa_start. The queue manager then becomes involved in the unit of work only if its resources are updated within sync point control. If this occurs, the extended transactional client calls ax_reg to register the queue manager's involvement.

Using the extended transactional client with SSL channels

You cannot set up an SSL channel using the xa_open string. Follow these instructions to use the client channel definition table (ccdt).

About this task

Because of the limited size of the xa_open xa_info string, it is not possible to pass all the information required to set up an SSL channel using the xa_open string method of connecting to a queue manager. Therefore you must either use the client channel definition table or, if your transaction manager allows, create the channel with MQCONNX before issuing the xa_open call.

To use the client channel definition table, follow these steps:

Procedure

1. Specify an xa_open string containing only the mandatory qmname (queue manager name) parameter, for example: `XA_Open_String=qmname=MYQM`
2. Use a queue manager to define a CLNTCONN (client-connection) channel with the required SSL parameters. Include the queue manager name in the QMNAME attribute on the CLNTCONN definition. This will be matched up with the qmname in the xa_open string.
3. Make the CLNTCONN definition available to the client system in a client channel definition table (CCDT) or, on Windows, in the active directory.
4. If you are using a CCDT, identify the CCDT containing the definition of the CLNTCONN channel using environment variables MQCHLLIB and MQCHLTAB. Set these variables in the environments used by both the client application and the transaction manager.

Results

This gives the transaction manager a channel definition to the appropriate queue manager with the SSL attributes needed to authenticate correctly, including SSLCIPH, the CipherSpec.

Configuring an extended transactional client for CICS

You configure an extended transactional client for use by CICS by adding an XAD resource definition to a CICS region.

Add the XAD resource definition by using the CICS resource definition online (RDO) command, **`cicsadd`**. The XAD resource definition specifies the following information:

- An xa_open string
- The fully qualified path name of a switch load file

One switch load file is supplied for use by CICS on each of the following platforms: AIX, HP-UX, Solaris, and Windows systems. Each switch load file contains a function that returns a pointer to the XA switch structure that is used for dynamic registration, MQRMIXASwitchDynamic. See Table 6 on page 26 for the fully qualified path name of each switch load file.

Table 6. The switch load files

Platform	Switch load file
AIX HP-UX Linux Solaris	*MQ_INSTALLATION_PATH*/lib/amqczsc
Windows systems	*MQ_INSTALLATION_PATH*\bin\mqcc4swi.dll [1]

MQ_INSTALLATION_PATH represents the high-level directory in which IBM MQ is installed.

Here is an example of an XAD resource definition for Windows systems:

```
cicsadd -c xad -r REGION1 WMQXA \
    ResourceDescription="IBM MQ queue manager MARS" \
    XAOpen="channel=MARS.SVR,trptype=tcp,conname=MARS(1415),qmname=MARS,tpm=cics" \
    SwitchLoadFile="C:\Program Files\IBM\WebSphere MQ\bin\mqcc4swi.dll"
```

For more information about adding an XAD resource definition to a CICS region, see the *CICS Administration Reference* and the *CICS Administration Guide* for your platform.

Note the following information about using CICS with an extended transactional client:

- You can add only one XAD resource definition for IBM MQ to a CICS region. This means that only one queue manager can be associated with a region, and all CICS applications that run in the region can connect only to that queue manager. If you want to run CICS applications that connect to a different queue manager, you must run the applications in a different region.

- Each application server in a region calls xa_open while it is initializing and starts an MQI channel to the queue manager associated with the region. This means that the queue manager must be started before an application server starts, otherwise the xa_open call fails. All IBM MQ MQI client applications later processed by the application server use the same MQI channel.
- When an MQI channel starts, and there is no security exit at the client end of the channel, the user ID that flows from the client system to the server connection MCA is `cics`. Under certain circumstances, the queue manager uses this user ID for authority checks when the server connection MCA subsequently attempts to access the queue manager resources on behalf of a client application. If this user ID is used for authority checks, you must ensure that it has the authority to access all the resources it needs to access.

  For information about when the queue manager uses this user ID for authority checks, see Security.
- The CICS task termination exits that are supplied for use on IBM MQ client systems are listed in Table 7 on page 27. You configure these exits in the same way that you configure the corresponding exits for IBM MQ server systems. For this information, therefore, see the Enabling CICS user exits.

Table 7. CICS task termination exits

Platform	Source	Library
AIX HP-UX Linux Solaris	amqzscgx.c	amqczscg
Windows systems	amqzscgn.c	mqcc1415.dll

Configuring an extended transactional client for Tuxedo

To configure XAD resource definition for use by Tuxedo, update the UBBCONFIG file and resource manager table.

To configure XAD resource definition for use by Tuxedo, perform the following actions:

- In the GROUPS section of the Tuxedo UBBCONFIG file for an application, use the OPENINFO parameter to specify an xa_open string.

  For an example of how to do this, see the sample UBBCONFIG file, which is supplied for use with the Tuxedo sample programs. On AIX, HP-UX, and Solaris, the name of the file is ubbstxcx.cfg and, on Windows systems, the name of the file is `ubbstxcn.cfg`.
- In the entry for a queue manager in the Tuxedo resource manager table:
  - udataobj/RM (AIX, HP-UX, and Solaris)
  - udataobj\rm (Windows systems)

  specify the name of an XA switch structure and the fully qualified path name of the library that contains the structure. For an example of how to do this for each platform, see TUXEDO samples. Tuxedo supports dynamic registration of a resource manager, and so you can use either MQRMIXASwitch or MQRMIXASwitchDynamic.

Microsoft Transaction Server

No additional configuration is required before you can use MTS as a transaction manager. However, there are some points to note.

Note the following information about using MTS with the extended transactional client:

- An MTS application always starts an MQI channel when it connects to a server queue manager. MTS, in its role as a transaction manager, then uses the same MQI channel to communicate with the queue manager.

- Following a failure, MTS must be able to recover any incomplete units of work. To do this, MTS must be able to communicate with any queue manager that was participating in an incomplete unit of work at the time of the failure.

  When an MTS application connects to a server queue manager and starts an MQI channel, the extended transactional client extracts sufficient information from the parameters of the MQCONN or MQCONNX call to enable the channel to be restarted following a failure, if required. The extended transactional client passes the information to MTS, and MTS records the information in its log.

  If the MTS application issues an MQCONN call, this information is simply the name of the queue manager. If the MTS application issues an MQCONNX call and provides a channel definition structure, MQCD, the information also includes the name of the MQI channel, the network address of the server queue manager, and the communications protocol for the channel.

  In a recovery situation, MTS passes this information back to the extended transactional client, and the extended transactional client uses it to restart the MQI channel.

  If you ever need to change any configuration information, therefore, ensure that all incomplete units of work have been resolved before making the changes. Alternatively, ensure that the configuration changes do not affect the ability of the extended transactional client to restart an MQI channel using the information recorded by MTS. The following are examples of such configuration changes:

  - Changing the value of the MQSERVER environment variable
  - Changing entries in the client channel definition table (CCDT)
  - Deleting a server connection channel definition
- Note the following conditions when using an extended transactional client with MTS:
  - Within a single thread, a client application can be connected to only one queue manager at a time.
  - Each thread of a client application can connect to a different queue manager.
  - A client application cannot use shared connection handles.

Defining MQI channels

To create a new channel, you have to create **two** channel definitions, one for each end of the connection, using the same channel name and compatible channel types. In this case, the channel types are *server-connection* and *client-connection*.

User defined channels

When the server does not automatically define channels there are two ways of creating the channel definitions and giving the IBM MQ application on the IBM MQ MQI client machine access to the channel.

These two methods are described in detail:

1. Create one channel definition on the IBM MQ client and the other on the server.

   This applies to any combination of IBM MQ MQI client and server platforms. Use it when you are getting started on the system, or to test your setup.

   See "Creating server-connection and client-connection definitions on different platforms" on page 34 for details on how to use this method.
2. Create both channel definitions on the server machine.

   Use this method when you are setting up multiple channels and IBM MQ MQI client machines at the same time.

   See "Creating server-connection and client-connection definitions on the server" on page 37 for details on how to use this method.

Automatically defined channels

IBM MQ products on platforms other than z/OS include a feature that can automatically create a channel definition on the server if one does not exist.

If an inbound attach request is received from a client and an appropriate server-connection definition cannot be found on that queue manager, IBM MQ creates a definition automatically and adds it to the queue manager. The automatic definition is based on the definition of the default server-connection channel SYSTEM.AUTO.SVRCONN. You enable automatic definition of server-connection definitions by updating the queue manager object using the ALTER QMGR command with the CHAD parameter (or the PCF command Change Queue Manager with the ChannelAutoDef parameter).

Related concepts

"Channel control function" on page 152
The channel control function provides facilities for you to define, monitor, and control channels.

Windows Linux UNIX Creating and using AMQP channels

When you install the IBM MQ support for MQ Light APIs into your IBM MQ installation, you can run IBM MQ MQSC commands (**runmqsc**) to define, alter, delete, start, and stop a channel. You can also view the status of a channel.

Before you begin

This task assumes that you have installed the AMQP channel. You do this by selecting the AMQP Service component when installing IBM MQ. For more information, follow the link for your platform then find the table row for "AMQP Service":

- AIX IBM MQ components for AIX systems
- HP-UX IBM MQ components for HP-UX systems
- Linux IBM MQ rpm components for Linux systems
- Solaris IBM MQ components for Solaris systems
- Windows IBM MQ features for Windows systems

To make a test connection to the queue manager, you must have an MQ Light client. MQ Light clients are available for Node.js, Ruby, Java, and Python. For more information on available clients, see the IBM MQ Light community website.

This task is based on the MQ Light Node.js client. However, the steps relating to the IBM MQ queue manager are the same for any client.

About this task

The following procedure assumes that you have an existing non-production queue manager, upgraded to command level 801. To enable the 801 command level for your queue manager, you can run the following command:

```
strmqm -e CMDLEVEL=801 <QMNAME>
```

You must restart the queue manager after running this command. If you require a new queue manager, a sample script is included, which is located in the `<mqinstall>/amqp/samples` directory. The script creates a new queue manager at command level 801, starts the AMQP service, creates a new channel called `SAMPLE.AMQP.CHANNEL`, and starts the channel. If you run the sample script, either `SampleMQM.sh` on Linux, or `SampleMQM.bat` on Windows, you can start the following procedure at Step 6.

Note: AMQP channels do not support user defined AMQP services. AMQP channels only support the system default SYSTEM.AMQP.SERVICE service.

If you upgrade an existing queue manager to command level 801, a new default channel object is created. The default channel is called SYSTEM.DEF.AMQP. You can use the default channel to test MQ Light connections to the queue manager or you can create a new channel.

The following procedure uses the default channel.

Procedure

1. Start **runmqsc** from the *<mqinstall>*/bin/ directory:

```
runmqsc <QMNAME>
```

2. Check that the AMQP function is installed and working correctly.

   Use the **START SERVICE** command to start the IBM MQ service, which controls the JVM:

```
START SERVICE(SYSTEM.AMQP.SERVICE)
```

3. Set the MCAUSER user ID.

   When an AMQP client connects to a channel, the channel specifies an MCAUSER user ID, which is used on connections to the queue manager. The default value of MCAUSER is blank. Before any AMQP clients can connect to the queue manager you must specify an MCAUSER value, which must be a valid IBM MQ user who is authorized to publish and subscribe on IBM MQ topics.

   a) Use the **ALTER CHANNEL** command to set the MCAUSER user ID:

```
ALTER CHANNEL(SYSTEM.DEF.AMQP) CHLTYPE(AMQP) MCAUSER(User ID)
```

   b) Use the following two **setmqaut** commands to authorize your MCAUSER user ID to publish and subscribe to topics:

```
setmqaut -m <QMNAME> -t topic -n SYSTEM.BASE.TOPIC -p <MCAUSER>
 -all +pub +sub
```

   and

```
setmqaut -m <QMNAME> -t qmgr -p <MCAUSER> -all +connect
```

   If the channel is running while the MCAUSER user ID is added or altered, you must stop and restart the channel.

   Note: If the MCAUSER user ID is not set, or the MCAUSER user ID is not authorized to publish or subscribe to IBM MQ topics, you will receive an error message in the AMQP client.

4. Use the **START CHANNEL** command to start the default SYSTEM.DEF.AMQP channel:

```
START CHANNEL(SYSTEM.DEF.AMQP)
```

5. If you want to check the channel status, use the **DISPLAY CHSTATUS** command:

```
DISPLAY CHSTATUS(SYSTEM.DEF.AMQP) CHLTYPE(AMQP)
```

   When the channel is running correctly, STATUS(RUNNING) is displayed in the command output.

6. Change the default port.

   The default port for AMQP 1.0 connections is 5672. If you are already using port 5672, which is possible if you previously installed MQ Light, you need to change the port that your AMQP channel uses. Use the **ALTER CHANNEL** command to change the port:

```
ALTER CHANNEL(SYSTEM.DEF.AMQP) CHLTYPE(AMQP) PORT(NEW PORT NUMBER)
```

7. If you do not want to block or filter connections to the AMQP channel using channel authentication (CHLAUTH) rules, disable channel authentication on the queue manager as follows:

```
alter qmgr chlauth(disabled)
```

You are not recommended to disable connection authentication on a production queue manager. You should only disable connection authentication in a development environment.

Alternatively, configure the queue manager channel authentication rules to allow specific connections to the AMQP channel.

8. Optional: If you want to enable SSL/TLS encryption on the channel, using the configured key repository for the queue manager, you must set the SSLCIPH attribute for the channel to an appropriate cipher specification. By default, the cipher specification is blank, meaning that SSL/TLS encryption is not used on the channel. Use the **ALTER CHANNEL** command to set a cipher specification. For example:

   ```
   ALTER CHANNEL(SYSTEM.DEF.AMQP) CHLTYPE(AMQP) SSLCIPH(CIPHER SPECIFICATION)
   ```

   Additionally, there are a number of other channel configuration options associated with SSL/TLS encryption that you can set as follows:

   - By default, the certificate in the queue manager key repository with label corresponding to the queue manager CERTLABL attribute is the name used by the SSL/TLS encryption for the channel. You can select a different certificate by setting CERTLABL. Use the **ALTER CHANNEL** command to specify the label for the required certificate:

     ```
     ALTER CHANNEL(SYSTEM.DEF.AMQP) CHLTYPE(AMQP) CERTLABL(CERTIFICATE LABEL)
     ```

   - You can set the channel to require a certificate from SSL/TLS client connections. You can select whether a certificate is required from a SSL/TLS client connection by setting SSLCAUTH. Use the **ALTER CHANNEL** command to set whether a certificate is required from a SSL/TLS client connection. For example:

     ```
     ALTER CHANNEL(SYSTEM.DEF.AMQP) CHLTYPE(AMQP) SSLCAUTH(REQUIRED or OPTIONAL)
     ```

   - V 8.0.0.15 If you set the SSLCAUTH attribute to REQUIRED, the Distinguished Name (DN) of the certificate from the client can be checked. To check the Distinguished Name of the certificate from the client set the SSLPEER attribute. Use the **ALTER CHANNEL** command to check the Distinguished Name of the certificate from the client. For example:

     ```
     ALTER CHANNEL(SYSTEM.DEF.AMQP) CHLTYPE(AMQP) SSLPEER (DN SPECIFICATION)
     ```

     Alternatively, you can also use channel authentication records to allow or block connections because this method offers greater granularity compared to using the SSLPEER attribute. For more information on setting SSLPEER and using channel authentication records as an alternative, see SSL Peer.

9. Install the MQ Light Node.js client by running the following command:

   ```
   npm install mqlight
   ```

10. Navigate to the `node_modules/mqlight/samples` directory, and run the sample receiver application:

    - If you are using the default port number, you can run the sample receiver application:

      ```
      node recv.js
      ```

    - If you configured your AMQP channel to use a different port number, you can run the sample receiver application with a parameter to specify the new port number:

```
node recv.js -s amqp://localhost:6789
```

A successful connection to the default channel displays the following message:

```
Connected to amqp://localhost:5672 using client-id recv_e79c55d
Subscribed to pattern: public
```

The application is now connected to the queue manager, and is waiting to receiving messages. It is subscribed to the topic `public`.

Note: The `client-id` is automatically generated, unless you specify one using the `-i` parameter.

11. In a new command window, navigate to the `node_modules/mqlight/samples` directory, and run the sample sender application by running the following command:

```
node send.js
```

In the command window for the receiver application, the `Hello World` message is displayed.

12. Use the **AMQSSUB** IBM MQ sample to receive an MQ Light sample message.

On Linux and Windows, the sample can be found in the following locations:

- Linux *mqinstall*/samp/bin directory on Linux.
- Windows *mqinstall*/Tools\c\Samples\Bin directory on Windows.

a) Run the sample by running the following command:

```
amqssub public <QM-name>.
```

b) Send a message to the IBM MQ application by re-running the following command:

```
node send.js
```

13. Use the **DEFINE CHANNEL** command to create more AMQP channels:

```
DEFINE CHANNEL(MY.AMQP.CHANNEL) CHLTYPE(AMQP) PORT(2345)
```

When you define a channel, it must be manually started, using the **START CHANNEL** command:

```
START CHANNEL(MY.AMQP.CHANNEL)
```

To check that the channel is running correctly you can run the sample receiver application, specifying the port of the new channel:

```
node recv.js -s amqp://localhost:2345
```

What to do next

You can use the following commands to display the IBM MQ connections, stop the channel, and delete the channel:

DISPLAY CONN(*) TYPE(*CONN*) WHERE (*CHANNEL EQ SYSTEM.DEF.AMQP*)
Displays the IBM MQ connection that the AMQP channel made on the queue manager.

DISPLAY CHSTATUS(*) CHLTYPE(*AMQP*) CLIENTID(*) ALL
Displays a list of the AMQP clients connected to the specified channel.

STOP CHANNEL (*MY.AMQP.CHANNEL*)
Stops an AMQP channel, and closes the port that it is listening on.

DELETE CHANNEL (*MY.AMQP.CHANNEL*)
Deletes any channels that you created.

Note: Do not delete the default channel SYSTEM.DEF.AMQP.

You can determine whether the AMQP capability is installed into the IBM MQ installation, and whether there is a queue manager associated with it, by using either **runmqsc** or PCF:

- Using **runmqsc**, display the attributes of the queue manager and check for `AMQPCAP(YES)`.
- Using PCF, use the **MQCMD_INQUIRE_Q_MGR** command, and confirm the value of `MQIA_AMQP_CAPABILITY`.

Related tasks
Developing AMQP client applications
Securing AMQP clients
Related reference
strmqm

Windows Linux UNIX Removing the AMQP channel from queue managers

You can remove the AMQP channel from queue managers by removing folders from the installation directory.

Procedure

1. Stop the queue manager.
2. Remove the IBM MQ support for MQ Light APIs:
   - AIX On AIX, run the following command:

     ```
     installp -u mqm.amqp.rte
     ```

   - Linux On Linux, remove the AMQP RPM. If you repackaged the RPM before you installed it, specify the name of the repackaged RPM.

     ```
     rpm -e MQSeriesAMQP
     ```

   - Windows On Windows, remove the amqp folder from the IBM MQ installation. Ensure that no other files or folders in the IBM MQ installation path are removed.
3. Restart the queue manager.

Related tasks
Developing AMQP client applications
Securing AMQP clients

Windows Linux UNIX AMQP channel log files

The log files for AMQP channels are stored in the same IBM MQ data directory as IBM MQ log files.

The default data directory on Windows is `C:\ProgramData\IBM\MQ`.

The default data directory on Linux is `/var/mqm`.

The AMQP channel writes log information to the following log files, found in the IBM MQ data directory:

- `amqp.stdout`, written to the `qmgrs/`<*QM-name*> folder.
- `amqp.stderr`, written to the `qmgrs/`<*QM-name*> folder.

- `amqp_*.log` , written to the `qmgrs/<QM-name>/errors` folder.

If an MQ Light client receives an authentication or authorization error, your administrator can find detailed information about the reason for the security failure in the `amqp_0.log` file and the `MQ AMQERR*.log` files.

Any FDC files are created as `AMQP*.FDC` files, which are written to the `<data-directory>/errors` folder.

Some configuration files are written to the `qmgrs/<QM-name>/amqp` directory. You do not need to edit any of the files in this directory.

Related concepts

Error logs on UNIX, Linux, and Windows

Related tasks

Developing AMQP client applications

Securing AMQP clients

Creating server-connection and client-connection definitions on different platforms

You can create each channel definition on the computer to which it applies. There are restrictions on how you can create channel definitions on a client computer.

On all platforms, you can use IBM MQ Script (MQSC) commands, programmable command format (PCF) commands, or the IBM MQ Explorer to define a server-connection channel on the server machine. z/OS On z/OS you can also use the Operation and Control panels. IBM i On IBM i you can also use the panel interface.

Because MQSC commands are not available on a machine where IBM MQ has been installed as an IBM MQ MQI client only, you must use different ways of defining a client-connection channel on the client machine.

Using **runmqsc**

You can specify the **-c** parameter and, optionally, the **-u** parameter to connect **runmqsc** as a client to the queue manager you want to administer.

If you use the **-u** parameter to supply a user ID, you are prompted for a matching password.

If you have configured the `CONNAUTH AUTHINFO` record with `CHCKLOCL(REQUIRED)` or `CHCKLOCL(REQDADM)`, you must use the **-u** parameter otherwise you will not be able to administer your queue manager with **runmqsc**.

Related concepts

"Creating a client-connection channel on the IBM MQ MQI client" on page 35
You can define a client-connection channel on the client workstation using MQSERVER or using the MQCNO structure on an MQCONNX call.

Related tasks

"Defining a server-connection channel on the server" on page 34
Start MQSC if necessary, then define the server-connection channel.

Defining a server-connection channel on the server

Start MQSC if necessary, then define the server-connection channel.

Procedure

1. Optional: If your server platform is not z/OS, first create and start a queue manager and then start MQSC commands.

a) Create a queue manager, called QM1 for example:

```
crtmqm QM1
```

b) Start the queue manager:

```
strmqm QM1
```

c) Start MQSC commands:

```
runmqsc QM1
```

2. Define a channel with your chosen name and a channel type of *server-connection*.

```
DEFINE CHANNEL(CHAN1) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server-connection to Client_1')
```

   This channel definition is associated with the queue manager running on the server.

3. Use the following command to allow the inbound connect access to your queue manager:

```
SET CHLAUTH(CHAN1) TYPE(ADDRESSMAP) ADDRESS('IP address') MCAUSER('userid')
```

   - Where SET CHLAUTH uses the name of the channel defined in the previous step.
   - Where *'IP address'* is the IP address of the client.
   - Where *'userid'* is the ID you want to provide to the channel for access control to the target queues. This field is case-sensitive.

   You can choose to identify your inbound connection using a number of different attributes. The example uses IP address. Alternative attributes include client user ID and SSL or TLS Subject Distinguished Name. For more information, see Channel authentication records

Creating a client-connection channel on the IBM MQ MQI client

You can define a client-connection channel on the client workstation using MQSERVER or using the MQCNO structure on an MQCONNX call.

Using MQSERVER

You can use the MQSERVER environment variable to specify a simple definition of a client-connection channel. It is simple in the sense that you can specify only a few attributes of the channel using this method.

- Specify a simple channel definition on Windows as follows:

```
SET MQSERVER=ChannelName/TransportType/ConnectionName
```

- Specify a simple channel definition on UNIX and Linux systems as follows:

```
export MQSERVER=ChannelName/TransportType/ConnectionName
```

- Specify a simple channel definition on IBM i systems as follows:

```
ADDENVVAR ENVVAR(MQSERVER) VALUE('ChannelName/TransportType/ConnectionName')
```

where:

- `ChannelName` must be the same name as defined on the server. It cannot contain a forward slash.

- `TransportType` can be one of the following values, depending on your IBM MQ MQI client platform:
  - LU62
  - TCP
  - NETBIOS
  - SPX

  Note: On UNIX and Linux systems, the TransportType is case-sensitive and must be uppercase. An `MQCONN` or `MQCONNX` call returns 2058 if the TransportType is not recognized
- `ConnectionName` is the name of the server as defined to the communications protocol (TransportType).

For example, on Windows:

```
SET MQSERVER=CHANNEL1/TCP/MCID66499
```

or, on UNIX and Linux systems:

```
export MQSERVER=CHANNEL1/TCP/'MCID66499'
```

Note: To change the TCP/IP port number, see "MQSERVER" on page 75.

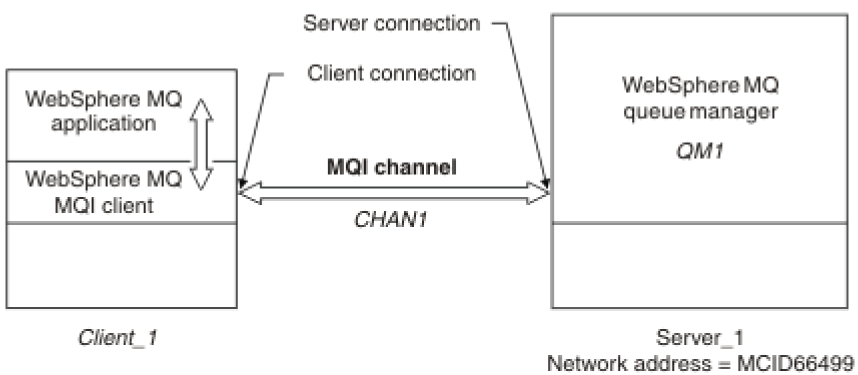

Figure 1. Simple channel definition

Some more examples of simple channel definitions are:

- On Windows:

```
SET MQSERVER=CHANNEL1/TCP/9.20.4.56
SET MQSERVER=CHANNEL1/NETBIOS/BOX643
```

- On UNIX and Linux systems:

```
export MQSERVER=CHANNEL1/TCP/'9.20.4.56'
export MQSERVER=CHANNEL1/LU62/BOX99
```

- IBM i On IBM i:

```
ADDENVVAR ENVVAR(MQSERVER) VALUE('CHANNEL1/TCP/9.20.4.56(1416)')
```

where BOX99 is the LU 6.2 ConnectionName.

On the IBM MQ MQI client, all **MQCONN** or **MQCONNX** requests then attempt to use the channel you have defined, unless the channel is overridden in an MQCD structure referenced from the MQCNO structure supplied to **MQCONNX**.

Note: For more information about the *MQSERVER* environment variable, see "MQSERVER" on page 75.

Using the MQCNO structure on an MQCONNX call

An IBM MQ MQI client application can use the connect options structure, MQCNO, on an **MQCONNX** call to reference a channel definition structure, MQCD, that contains the definition of a client-connection channel.

In this way, the client application can specify the **ChannelName**, **TransportType**, and **ConnectionName** attributes of a channel at run time, enabling the client application to connect to multiple server queue managers simultaneously.

Note that if you define a channel using the *MQSERVER* environment variable, it is not possible to specify the **ChannelName**, **TransportType**, and **ConnectionName** attributes at run time.

A client application can also specify attributes of a channel such as **MaxMsgLength** and **SecurityExit**. Specifying such attributes enables the client application to specify values for the attributes that are not the default values, and enables channel exit programs to be called at the client end of an MQI channel.

If a channel uses the Secure Sockets Layer (SSL) or Transport Layer Security (TLS), a client application can also provide information relating to SSL or TLS in the MQCD structure. Additional information relating to SSL or TLS can be provided in the SSL or TLS configuration options structure, MQSCO, which is also referenced by the MQCNO structure on an **MQCONNX** call.

For more information about the MQCNO, MQCD, and MQSCO structures, see MQCNO, MQCD, and MQSCO.

Note: The sample program for MQCONNX is called **amqscnxc**. Another sample program called **amqssslc** demonstrates use of the MQSCO structure.

Creating server-connection and client-connection definitions on the server

You can create both definitions on the server, then make the client-connection definition available to the client.

First define a server-connection channel and then define a client-connection channel. On all platforms, you can use IBM MQ Script (MQSC) commands, programmable command format (PCF) commands or the IBM MQ Explorer to define a server-connection channel on the server machine. z/OS On z/OS you can also use the Operation and Control panels. IBM i On IBM i you can also use the panel interface.

Client-connection channel definitions created on the server are made available to clients using a client channel definition table (CCDT).

Related concepts

"Client channel definition table" on page 38
The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On platforms other than z/OS a CCDT is created automatically. You must then make it available to the client application.

Related tasks

"Defining the server-connection channel on the server" on page 40
Create a server-connection channel definition for the queue manager.

"Defining the client-connection channel on the server" on page 41
Having defined the server-connection channel, you now define the corresponding client-connection channel.

"Accessing client-connection channel definitions" on page 42

Make the client channel definition table (CCDT) available to client applications by copying or sharing it, then specify its location and name on the client computer.

Client channel definition table

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On platforms other than z/OS a CCDT is created automatically. You must then make it available to the client application.

The purpose of the client channel definition table (CCDT) is to determine the channel definitions used by client applications to connect to the queue manager. The channel definition also specifies the authentication information that applies to the connections.

The CCDT is a binary file. It is generated by a queue manager. The queue manager does not read the CCDT file.

On platforms other than z/OS, the CCDT is created when the queue manager is created. Client connection channels are added to the table when you use the **DEFINE CHANNEL** command, and their definitions altered when you issue the **ALTER CHANNEL** command.

Notes:

- The design of the IBM MQ CCDT file, is that the CCDT file is shrunk, only after all client-connection channels defined by the user are actually defined. When a client-connection channel is deleted, it is just marked as deleted in the CCDT file, but it is not physically removed.
- To force the CCDT file to shrink, after deleting one or more client-connection channels, issue the following command:

```
rcrmqobj -m QM80 -t clchltab
```

You can use the CCDT to provide clients with the authentication information to check for SSL certificate revocation. Define a namelist containing authentication information objects and set the queue manager attribute **SSLCRLNameList** to the name of the namelist.

There are a number of ways for a client application to use a CCDT. The CCDT can be copied to the client computer. You can copy the CCDT to a location shared by more than one client. You can make the CCDT accessible to the client as a shared file, while it remains located on the server.

If you use FTP to copy the file, use the `bin` option to set binary mode; do not use the default `ASCII` mode. Whichever method you choose to make the CCDT available, the location must be secure to prevent unauthorized changes to the channels.

Server platforms other than z/OS

A default CCDT called `AMQCLCHL.TAB` is created when you create a queue manager.

By default, AMQCLCHL.TAB is located in the following directory on a server:

- IBM i On IBM i, in the integrated file system:

```
/QIBM/UserData/mqm/qmgrs/QUEUEMANAGERNAME/&ipcc
```

- Linux UNIX On UNIX and Linux systems:

```
/prefix/qmgrs/QUEUEMANAGERNAME/@ipcc
```

The name of the directory referenced by *QUEUEMANAGERNAME* is case-sensitive on UNIX and Linux systems. The directory name might not be the same as the queue manager name, if the queue manager name has special characters in it.

- Windows On Windows:

```
MQ_INSTALLATION_PATH\data\qmgrs\QUEUEMANAGERNAME\@ipcc
```

MQ_INSTALLATION_PATH represents the high-level directory in which IBM MQ is installed.

However, you might have chosen to use a different directory for queue manager data. You can specify the parameter **-md** *DataPath* when you used the **crtmqm** command. If you do, AMQCLCHL.TAB is located in the @ipcc directory of the *DataPath* you specified.

The path to the CCDT can be changed by setting MQCHLLIB. If you do set MQCHLLIB, be aware, if you have multiple queue managers on the same server, they share the same CCDT location.

The CCDT is created when the queue manager is created. Each entry of a CCDT represents a client connection to a specific queue manager. A new entry is added when you define a client-connection channel using the **DEFINE CHANNEL** command, and the entry is updated when you alter the client-connection channels by using the **ALTER CHANNEL** command.

Client platforms at IBM MQ 8.0

You can create a CCDT on the client machine directly by using the runmqsc command with the **-n** parameter. The CCDT will be created in the location indicated by MQCHLLIB and with the filename indicated by MQCHLTAB which is AMQCLCHL.TAB by default.

Note, that if you specify the **-n** parameter, you must not specify any other parameter.

Each entry of a CCDT represents a client connection to a specific queue manager. A new entry is added when you define a client-connection channel using the **DEFINE CHANNEL** command, and the entry is updated when you alter the client-connection channels by using the **ALTER CHANNEL** command.

How to specify the location of the CCDT on the client

On a client system, you can specify the location of the CCDT in two ways:

- Using the environment variables MQCHLLIB to specify the directory where the table is located, and MQCHLTAB to specify the file name of the table.
- Using the client configuration file. In the CHANNELS stanza, use the attributes ChannelDefinitionDirectory to specify the directory where the table is located, and ChannelDefinitionFile to specify the file name.

If the location is specified both in the client configuration file and by using environment variables, the environment variables take priority. You can use this feature to specify a standard location in the client configuration file and override it using environment variables when necessary.

Related concepts

Working with revoked certificates

Related reference

"MQCHLLIB" on page 72

MQCHLLIB specifies the directory path to the file containing the client channel definition table (CCDT). The file is created on the server, but can be copied across to the IBM MQ MQI client workstation.

Migration and client channel definition tables (CCDT)

In general, the internal format of the client channel definition table might change from one release level of IBM MQ to the next. As a result, an IBM MQ MQI client can use a client channel definition table only when it has been prepared by a server queue manager that is at the same release level as the client, or at an earlier release level.

A Version 7.1 IBM MQ MQI client can use a client channel definition table that has been prepared by a Version 6.0 queue manager. But a Version 6.0 client cannot use a client channel definition table that has been prepared by a Version 7.1 queue manager.

Client connection channels in the Active Directory

On Windows systems that support the Active Directory, IBM MQ publishes client connection channels in the Active Directory to provide dynamic client-server binding.

When client connection channel objects are defined, they are written into a client channel definition file, called AMQCLCHL.TAB by default. If the client connection channels use the TCP/IP protocol, the IBM MQ server also publishes them in the Active Directory. When the IBM MQ client determines how to connect to the server, it looks for a relevant client connection channel object definition using the following search order:

1. MQCONNX MQCD data structure
2. MQSERVER environment variable
3. client channel definition file
4. Active Directory

This order means that any current applications are not affected by any change. You can think of these entries in the Active Directory as records in the client channel definition file, and the IBM MQ client processes them in the same way. To configure and administer support for publishing client connection channel definitions in the Active Directory, use the `setmqscp` command, as described in setmqscp.

Defining the server-connection channel on the server

Create a server-connection channel definition for the queue manager.

Procedure

1. On the server machine, define a channel with your chosen name and a channel type of *server-connection*.
   For example:

   ```
   DEFINE CHANNEL(CHAN2) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Server-connection to Client_2')
   ```

2. Use the following command to allow the inbound connect access to your queue manager:

   ```
   SET CHLAUTH(CHAN2) TYPE(ADDRESSMAP) ADDRESS('IP address') MCAUSER('userid')
   ```

   - Where SET CHLAUTH uses the name of the channel defined in the previous step.
   - Where *'IP address'* IP address is the IP address of the client.
   - Where *'userid'* is the ID you want to provide to the channel for access control to the target queues. This field is case-sensitive.

   You can choose to identify your inbound connection using a number of different attributes. The example uses IP address. Alternative attributes include client user ID and SSL or TLS Subject Distinguished Name. For more information, see Channel authentication records

   This channel definition is associated with the queue manager running on the server.

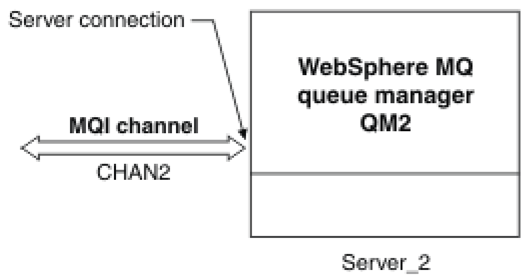

Figure 2. Defining the server-connection channel

Defining the client-connection channel on the server

Having defined the server-connection channel, you now define the corresponding client-connection channel.

Before you begin

Define the server-connection channel.

Procedure

1. Define a channel with the same name as the server-connection channel, but a channel type of *client-connection*. You must state the connection name (CONNAME). For TCP/IP, the connection name is the network address or host name of the server machine. It is also advisable to specify the queue manager name (QMNAME) to which you want your IBM MQ application, running in the client environment, to connect. By varying the queue manager name, you can define a set of channels to connect to different queue managers.

```
DEFINE CHANNEL(CHAN2) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) QMNAME(QM2) DESCR('Client-connection to Server_2')
```

2. Use the following command to allow the inbound connect access to your queue manager:

```
SET CHLAUTH(CHAN2) TYPE(ADDRESSMAP) ADDRESS('IP-address') MCAUSER('userid')
```

   - Where SET CHLAUTH uses the name of the channel defined in the previous step.
   - Where *'IP address'* is the IP address of the client.
   - Where *'userid'* is the ID you want to provide to the channel for access control to the target queues. This field is case-sensitive.

   You can choose to identify your inbound connection using a number of different attributes. The example uses IP address. Alternative attributes include client user ID and SSL or TLS Subject Distinguished Name. For more information, see Channel authentication records

Results

On platforms other than z/OS, this channel definition is stored in a file called the client channel definition table (CCDT), which is associated with the queue manager. The client channel definition table can contain more than one client-connection channel definition. For more information about the client channel definition table, and for the corresponding information about how client-connection channel definitions are stored on z/OS, see “Client channel definition table” on page 38.

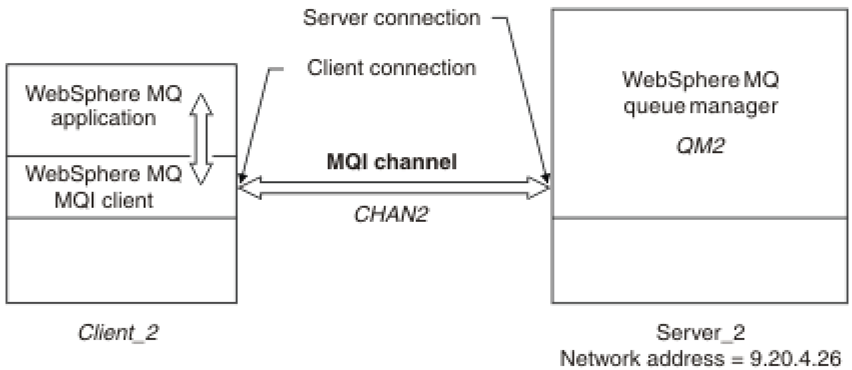

Figure 3. Defining the client-connection channel

Accessing client-connection channel definitions

Make the client channel definition table (CCDT) available to client applications by copying or sharing it, then specify its location and name on the client computer.

Before you begin

You have defined the client-connection channels you need.

On z/OS, you have created a CCDT. On other platforms, the CCDT is automatically created and updated.

About this task

For a client application to use the client channel definition table (CCDT), you must make the CCDT available to it and specify its location and name

Procedure

1. Make the CCDT available to the client applications in one of three ways:
   a) Optional: Copy the CCDT to the client computer.
   b) Optional: Copy the CCDT to a location shared by more than one client.
   c) Optional: Leave the CCDT on the server but make it shareable by the client.

   Whichever location you choose for the CCDT, the location must be secure to prevent unauthorized changes to the channels.
2. On the client, specify the location and name of the file containing the CCDT in one of three ways:
   a) Optional: Use the CHANNELS stanza of the client configuration file. For more information, see "CHANNELS stanza of the client configuration file" on page 58.
   b) Optional: Use the environment variables MQCHLLIB and MQCHLTAB.

      For example, you can set the environment variables by typing:

      - On HP Integrity NonStop Server, and UNIX and Linux systems:

```
export MQCHLLIB= MQ_INSTALLATION_PATH/qmgrs/ QUEUEMANAGERNAME /@ipcc
export MQCHLTAB=AMQCLCHL.TAB
```

      - IBM i On IBM i:

```
ADDENVVAR ENVVAR(MQCHLLIB) VALUE('/QIBM/UserData/mqm/qmgrs/QUEUEMANAGERNAME/@ipcc')
ADDENVVAR ENVVAR(MQCHLTAB) VALUE(AMQCLCHL.TAB)
```

      where *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.

c) Optional: On Windows only, use the **setmqscp** control command to publish the client-connection channel definitions in Active Directory

If the MQSERVER environment variable is set, an IBM MQ client uses the client-connection channel definition specified by MQSERVER in preference to any definitions in the client channel definition table.

Channel-exit programs for MQI channels

Three types of channel exit are available to the IBM MQ MQI client environment on UNIX, Linux and Windows systems.

These are:

- Send exit
- Receive exit
- Security exit

These exits are available at both the client and the server end of the channel. Exits are not available to your application if you are using the MQSERVER environment variable. Channel exits are explained in Channel exit programs for messaging channels.

The send and receive exits work together. There are several possible ways in which you can use them:

- Splitting and reassembling a message
- Compressing and decompressing data in a message (this functionality is provided as part of IBM MQ, but you might want to use a different compression technique)
- Encrypting and decrypting user data (this functionality is provided as part of IBM MQ, but you might want to use a different encryption technique)
- Journaling each message sent and received

You can use the security exit to ensure that the IBM MQ client and server are correctly identified, and to control access.

If send or receive exits on the server-connection side of the channel instance need to perform MQI calls on the connection with which they are associated, they use the connection handle provided in the `MQCXP` `Hconn` field. You must be aware that client-connection send and receive exits cannot make MQI calls.

Related concepts

"Security exits on a client connection" on page 44
You can use security exit programs to verify that the partner at the other end of a channel is genuine. Special considerations apply when a security exit is applied to a client connection.

Related tasks

Extending queue manager facilities

Related reference

"Path to exits" on page 44
A default path for location of the channel exits is defined in the client configuration file. Channel exits are loaded when a channel is initialized.

"Identifying the API call in a send or receive exit program" on page 45
When you use MQI channels for clients, byte 10 of the agent buffer identifies the API call in use when a send or receive exit is called. This is useful for identifying which channel flows include user data and might require processing such as encryption or digital signing.

User exits, API exits, and IBM MQ installable services

Path to exits

A default path for location of the channel exits is defined in the client configuration file. Channel exits are loaded when a channel is initialized.

On UNIX, Linux and Windows systems, a client configuration file is added to your system during installation of the IBM MQ MQI client. A default path for location of the channel exits on the client is defined in this file, using the stanza:

```
ClientExitPath:
ExitsDefaultPath= string
ExitsDefaultPath64= string
```

where *string* is a file location in a format appropriate to the platform

When a channel is initialized, after an `MQCONN` or `MQCONNX` call, the client configuration file is searched. The ClientExitPath stanza is read and any channel exits that are specified in the channel definition are loaded.

Security exits on a client connection

You can use security exit programs to verify that the partner at the other end of a channel is genuine. Special considerations apply when a security exit is applied to a client connection.

Figure 4 on page 45 illustrates the use of security exits in a client connection, using the IBM MQ object authority manager to authenticate a user. Either SecurityParmsPtr or SecurityParmsOffset is set in the MQCNO structure on the client and there are security exits at both ends of the channel. After the normal security message exchange has ended, and the channel is ready to run, the MQCSP structure accessed from the MQCXP SecurityParms field is passed to the security exit on the client. The exit type is set to MQXR_SEC_PARMS. The security exit can elect to do nothing to the user identifier and password, or it can alter either or both of them. The data returned from the exit is then sent to the server-connection end of the channel. The MQCSP structure is rebuilt on the server-connection end of the channel and is passed to the server-connection security exit accessed from the MQCXP SecurityParms field. The security exit receives and processes this data. This processing is typically to reverse any change made to the user ID and password fields in the client exit, which are then used to authorize the queue manager connection. The resulting MQCSP structure is referenced using SecurityParmsPtr in the MQCNO structure on the queue manager system.

The memory address that is passed back by the MQCXP SecurityParms field must remain addressable and unchanged until MQXR_TERM. An exit must not invalidate or free the memory back to the system before the exit is called for MQXR_TERM.

If SecurityParmsPtr or SecurityParmsOffset are set in the MQCNO structure and there is a security exit at only one end of the channel, the security exit receives and processes the MQCSP structure. Actions such as encryption are inappropriate for a single user exit, as there is no exit to perform the complementary action.

If SecurityParmsPtr and SecurityParmsOffset are not set in the MQCNO structure and there is a security exit at either or both ends of the channel, the security exit or exits are called. Either security exit can return its own MQCSP structure, addressed through the SecurityParmsPtr; the security exit is not called again until it is terminated (ExitReason of MQXR_TERM). The exit writer can free the memory used for the MQCSP at that stage.

When a server-connection channel instance is sharing more than one conversation, the pattern of calls to the security exit is restricted on the second and subsequent conversations.

For the first conversation, the pattern is the same as if the channel instance is not sharing conversations. For the second and subsequent conversations, the security exit is never called with MQXR_INIT, MQXR_INIT_SEC, or MQXR_SEC_MSG. It is called with MQXR_SEC_PARMS.

In a channel instance with sharing conversations, MQXR_TERM is called only for the last conversation running.

Each conversation has the opportunity in the MQXR_SEC_PARMS invocation of the exit to alter the MQCD; on the server-connection end of the channel this feature can be useful to vary, for example, the MCAUserIdentifier or LongMCAUserIdPtr values before the connection is made to the queue manager.

Server-connection exit	Client-connection exit
	Invoked with MQXR_INIT Responds with MQXCC_OK
Invoked with MQXR_INIT Responds with MQXCC_OK	
	Invoked with MQXR_INIT_SEC Responds with MQXCC_OK
Invoked with MQXR_INIT_SEC Responds with MQXCC_OK	
	Invoked with MQXR_SEC_PARMS Responds with MQXCC_OK
Invoked with MQXR_SEC_PARMS Responds with MQXCC_OK	
Data transfer begins	
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Figure 4. Client connection-initiated exchange with agreement for client connection using security parameters

Note: Security exit applications constructed prior to the release of IBM MQ v7.1 may require updating. For more information see Channel security exit programs.

Identifying the API call in a send or receive exit program

When you use MQI channels for clients, byte 10 of the agent buffer identifies the API call in use when a send or receive exit is called. This is useful for identifying which channel flows include user data and might require processing such as encryption or digital signing.

The following table shows the data that appears in byte 10 of the channel flow when an API call is being processed.

Note: These are not the only values of this byte. There are other **reserved** values.

Table 8. Identifying API calls

API call	Value of byte 10 for request	Value of byte 10 for reply
MQCONN “1” on page 46, “2” on page 46	X'81'	X'91'

Table 8. Identifying API calls (continued)		
API call	**Value of byte 10 for request**	**Value of byte 10 for reply**
MQDISC “1” on page 46	X'82'	X'92'
MQOPEN “3” on page 46	X'83'	X'93'
MQCLOSE	X'84'	X'94'
MQGET “4” on page 46	X'85'	X'95'
MQPUT “4” on page 46	X'86'	X'96'
MQPUT1 request “4” on page 46	X'87'	X'97'
MQSET request	X'88'	X'98'
MQINQ request	X'89'	X'99'
MQCMIT request	X'8A'	X'9A'
MQBACK request	X'8B'	X'9B'
MQSTAT request	X'8D'	X'9D'
MQSUB request	X'8E'	X'9E'
MQSUBRQ request	X'8F'	X'9F'
xa_start request	X'A1'	X'B1'
xa_end request	X'A2'	X'B2'
xa_open request	X'A3'	X'B3'
xa_close request	X'A4'	X'B4'
xa_prepare request	X'A5'	X'B5'
xa_commit request	X'A6'	X'B6'
xa_rollback request	X'A7'	X'B7'
xa_forget request	X'A8'	X'B8'
xa_recover request	X'A9'	X'B9'
xa_complete request	X'AA'	X'BA'

Notes:

1. The connection between the client and server is initiated by the client application using MQCONN. Therefore, for this command in particular, there are several other network flows. The same applies to MQDISC, which terminates the network connection.
2. MQCONNX is treated in the same way as MQCONN for the purposes of the client-server connection.
3. If a large distribution list is opened, there might be more than one network flow per MQOPEN call in order to pass all the required data to the SVRCONN MCA.
4. Large messages can exceed the transmission segment size. If this happens there can be many network flows resulting from a single API call.

z/OS Connecting a client to a queue-sharing group

You can connect a client to a queue-sharing group by creating an MQI channel between a client and a queue manager on a server that is a member of a queue-sharing group.

A queue-sharing group is formed by a set of queue-managers that can access the same set of shared queues. For more information about shared queues, see Shared queues and queue-sharing groups.

A client putting to a shared queue can connect to any member of the queue-sharing group. The benefits of connecting to a queue-sharing group are possible increases in front-end and back-end availability, and increased capacity. You can connect to a specific queue manager or to the generic interface.

Connecting directly to a queue manager in a queue-sharing group gives the benefit that you can put messages to a shared target queue, which increases back-end availability.

Connecting to the generic interface of a queue-sharing group opens a session with one of the queue managers in the group. This increases front-end availability, because the client queue manager can connect with any queue-manager in the group. You connect to the group using the generic interface when you do not want to connect to a specific queue manager within the queue-sharing group.

The generic interface can be a Sysplex Distributor VIPA address or a VTAM generic resource name, or another common interface to the queue-sharing group. For more details on setting up a generic interface, see Setting up communication for IBM MQ for z/OS using queue-sharing groups.

To connect to the generic interface of a queue-sharing group you need to create channel definitions that can be accessed by any queue manager in the group. To do this you need to have the same definitions on each queue manager in the group.

Define the SVRCONN channel as follows:

```
DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
QSGDISP(GROUP)
```

Channel definitions on the server are stored in a shared Db2® repository. Each queue manager in the queue-sharing group makes a local copy of the definition, ensuring that you will always connect to the correct server-connection channel when you issue an MQCONN or MQCONNX call.

Define the CLNTCONN channel as follows:

```
DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME( VIPA address ) QMNAME(QSG1) +
DESCR('Client-connection to Queue Sharing Group QSG1') QSGDISP(GROUP)
```

Because the generic interface of the queue-sharing group is stored in the CONNAME field in the client-connection channel, you can now connect to any queue manager in the group, and put to shared queues owned by that group.

Configuring a client using a configuration file

Configure your clients by using attributes in a text file. These attributes can be overridden by environment variables or in other platform-specific ways.

You can configure your IBM MQ MQI clients by using a text file, the IBM MQ MQI client configuration file. This file is similar to the queue manager configuration file, `qm.ini`, that is used on UNIX and Linux systems. The file contains a number of stanzas, each of which contains a number of lines of the format **`attribute-name`** = *value*.

The IBM MQ MQI client configuration file is generally named `mqclient.ini`, but you can choose to give it another name. The configuration information in this file applies to the following platforms:

- ULW UNIX, Linux, and Windows
- IBM i IBM i

Note: On IBM i, there is no default `mqclient.ini` file. However, you can create the file in the IBM i Integrated File System (IFS).

For more information, see "Location of the client configuration file" on page 49.

Note: z/OS The z/OS platform cannot be used to run IBM MQ clients. Therefore, the `mqclient.ini` file does not exist on IBM MQ for z/OS.

The attributes in the IBM MQ MQI client configuration file apply to clients that use:

- The MQI
- IBM MQ classes for Java
- IBM MQ classes for JMS
- IBM MQ classes for .NET
- XMS

Although the attributes in the IBM MQ MQI client configuration file apply to most IBM MQ clients, there are some attributes that are not read by managed .NET and XMS .NET clients, or by clients using either the IBM MQ classes for Java or the IBM MQ classes for JMS. For more information, see "Which IBM MQ clients can read each attribute" on page 50.

The configuration features apply to all connections a client application makes to any queue managers, rather than being specific to an individual connection to a queue manager. Attributes relating to a connection to an individual queue manager can be configured programmatically, for example by using an MQCD structure, or by using a Client Channel Definition Table (CCDT).

Environment variables that were supported in releases of IBM WebSphere MQ earlier than Version 7.0 continue to be supported, and where such an environment variable matches an equivalent value in the client configuration file, the environment variable overrides the client configuration file value.

For a client application that uses IBM MQ classes for JMS, you can also override the client configuration file in the following ways:

- By setting properties in the JMS configuration file
- By setting Java system properties, which also overrides the JMS configuration file.

For the .NET client, you can also override the client configuration file and the equivalent environment variables by using the .NET application configuration file.

You cannot set up multiple channel connections by using the client configuration file.

Example client configuration file

```
#* Module Name: mqclient.ini                                          *#
#* Type       : IBM MQ MQI client configuration file                  *#
#  Function   : Define the configuration of a client                  *#
#*                                                                    *#
#********************************************************************#
#* Notes      :                                                       *#
#* 1) This file defines the configuration of a client                 *#
#*                                                                    *#
#********************************************************************#

ClientExitPath:
   ExitsDefaultPath=/var/mqm/exits
   ExitsDefaultPath64=/var/mqm/exits64

TCP:
   Library1=DLLName1
   KeepAlive = Yes
   ClntSndBuffSize=32768
   ClntRcvBuffSize=32768
   Connect_Timeout=0

MessageBuffer:
   MaximumSize=-1
   Updatepercentage=-1
```

```
   PurgeTime=0

LU62:
   TPName
   Library1=DLLName1
   Library2=DLLName2

PreConnect:
    Module=amqldapi
    Function=myFunc
    Data=ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
    Sequence=1

CHANNELS:
DefRecon=YES
ServerConnectionParms=SALES.SVRCONN/TCP/hostname.x.com(1414)
```

Related reference

"Using IBM MQ environment variables" on page 71
This section describes the environment variables that you can use with IBM MQ MQI client applications.

"Changing queue manager configuration information" on page 103
The attributes described here modify the configuration of an individual queue manager. They override any settings for IBM MQ.

Location of the client configuration file

An IBM MQ MQI client configuration file can be held in a number of locations.

A client application uses the following search path to locate the IBM MQ MQI client configuration file:

1. The location specified by the environment variable MQCLNTCF.

   The format of this environment variable is a full URL. This means the file name might not necessarily be `mqclient.ini` and facilitates placing the file on a network attached file-system.

   Notes:

   - C, .NET and XMS clients support only the `file:` protocol; the `file:` protocol is assumed if the URL string does not begin with `protocol:`
   - To allow for Java 1.4.2 JREs, which do not support reading environment variables, the MQCLNTCF environment variable can be overridden with an MQCLNTCF Java System Property.
2. A file called `mqclient.ini` in the present working directory of the application.
3. A file called `mqclient.ini` in the IBM MQ data directory for Windows, UNIX and Linux systems.

   notes:

   - The IBM MQ data directory does not exist on certain platforms, for example, IBM i and z/OS, or in cases where the client has been supplied with another product.

     IBM i On IBM i, there is no default `mqclient.ini` file. However, the file can be created in the IBM i Integrated File System (IFS) in directory `/QIBM/UserData/mqm/`, and environment variable **MQCLNTCF** defined to point to it. For example:

     ```
     ADDENVVAR ENVVAR(MQCLNTCF) VALUE('QIBM/UserData/mqm/mqclient.ini') REPLACE(*YES)
     ```

     For more examples of environment variables, see Environment variables.

     z/OS The z/OS platform cannot be used to run IBM MQ clients. Therefore, the `mqclient.ini` file does not exist on IBM MQ for z/OS.
   - Linux UNIX On UNIX and Linux systems, the directory is `/var/mqm`
   - Windows On Windows platforms you configure the environment variable MQ_DATA_PATH during installation, to point at the data directory. It is normally `C:\ProgramData\IBM\MQ`

     Note: If you are installing a client only, the environment variable might be MQ_FILE_PATH.

- To allow for Java 1.4.2 JREs that do not support reading environment variables you can manually override the MQ_DATA_PATH environment variable with an MQ_DATA_PATH Java System Property.

4. A file called `mqclient.ini` in a standard directory appropriate to the platform, and accessible to users:
   - For all Java clients this is the value of the `user.home` Java System Property.
   - Linux UNIX For C clients on UNIX and Linux platforms this is the value of the HOME environment variable.
   - Windows For C clients on Windows this is the concatenated values of the HOMEDRIVE and HOMEPATH environment variables.

Note: For the IBM MQ client for HP Integrity NonStop Server, the `mqclient.ini` file must be located in the OSS file system. Guardian applications must either place the `mqclient.ini` file in the IBM MQ data directory or set the MQCLNTCF environment variable to a location in the OSS file system.

Which IBM MQ clients can read each attribute

Most of the attributes in the IBM MQ MQI client configuration file can be used by the C client, and the unmanaged .NET clients. However, there are some attributes that are not read by managed .NET and XMS .NET clients, or by clients using either the IBM MQ classes for Java or the IBM MQ classes for JMS.

Table 9. Which attributes apply to each type of client

mqclient.ini stanza name and attributes	**Description**	**C and unmanaged .NET**	**Java**	**JMS**	**Managed .NET**	**Managed XMS .NET**
CHANNELS stanza						
CCSID	The coded character set number to be used.	Yes	No	No	Yes	Yes
ChannelDefinitionDirectory	The directory path to the file containing the client channel definition table.	Yes	No	No	Yes	Yes
ChannelDefinitionFile	The name of the file containing the client channel definition table.	Yes	No	No	Yes	Yes

Table 9. Which attributes apply to each type of client (continued)						
mqclient.ini stanza name and attributes	**Description**	**C and unmanaged .NET**	**Java**	**JMS**	**Managed .NET**	**Managed XMS .NET**
ReconDelay	An administrative option to configure reconnect delay for client programs that can auto-reconnect.	Yes	No	Yes	Yes	Yes
DefRecon	An administrative option to enable client programs to automatically reconnect, or to disable the automatic reconnection of a client program that has been written to reconnect automatically.	Yes	No	Yes	Yes	Yes
MQReconnectTimeout	The timeout in seconds to reconnect to a client.	Yes	No	No	Yes	No
ServerConnectionParms	The location of the IBM MQ server and the communication method to be used.	Yes	No	No	Yes	Yes

Table 9. Which attributes apply to each type of client (continued)						
mqclient.ini stanza name and attributes	**Description**	**C and unmanaged .NET**	**Java**	**JMS**	**Managed .NET**	**Managed XMS .NET**
Put1DefaultAlwaysSync	Controls the behavior of the MQPUT1 function call with the option MQPMO_RESPONSE_AS_Q_DEF.	Yes	Yes	Yes	Yes	Yes
PasswordProtection	Allows you to set protected passwords in the MQCSP structure, rather than using SSL or TLS.	Yes	Yes	Yes	Yes	Yes
ClientExitPath stanza						
ExitsDefaultPath	Specifies the location of 32-bit channel exits for clients.	Yes	Yes	Yes	Yes	Yes
ExitsDefaultPath64	Specifies the location of 64-bit channel exits for clients.	Yes	Yes	Yes	Yes	Yes
JavaExitsClassPath	The values to be added to the classpath when a Java exit is run.	No	Yes	Yes	No	No
JMQI stanza						

Table 9. Which attributes apply to each type of client (continued)						
`mqclient.ini` stanza name and attributes	**Description**	**C and unmanaged .NET**	**Java**	**JMS**	**Managed .NET**	**Managed XMS .NET**
useMQCSPauthentication	Controls whether IBM MQ classes for Java and IBM MQ classes for JMS applications should use Compatibility mode or MQCSP authentication mode when authenticating with a queue manager.	No	Yes	Yes	No	No
MessageBuffer stanza						
MaximumSize	Size, in kilobytes, of the read-ahead buffer, in the range 1 through 999 999.	Yes	Yes	Yes	Yes	Yes
PurgeTime	Interval, in seconds, after which messages left in the read-ahead buffer are purged.	Yes	Yes	Yes	Yes	Yes
UpdatePercentage	The update percentage value, in the range of 1 - 100, used in calculating the threshold value to determine when a client application makes a new request to the server.	Yes	Yes	Yes	Yes	Yes

Table 9. Which attributes apply to each type of client (continued)

mqclient.ini stanza name and attributes	**Description**	**C and unmanaged .NET**	**Java**	**JMS**	**Managed .NET**	**Managed XMS .NET**
PreConnect stanza						
Data	URL of the repository where connection definitions are stored.	Yes	No	No	No	No
Function	Name of the functional entry point into the library that contains the PreConnect exit code.	Yes	No	No	No	No
Module	The name of the module containing the API exit code.	Yes	No	No	No	No
Sequence	The sequence in which this exit is called relative to other exits.	Yes	No	No	No	No
Security stanza						
DisableClientAMS	Disables or enables AMS for client connections to a queue manager.	Yes	Yes	Yes	No	No
SSL stanza						

Table 9. Which attributes apply to each type of client (continued)						
`mqclient.ini` stanza name and attributes	**Description**	**C and unmanaged .NET**	**Java**	**JMS**	**Managed .NET**	**Managed XMS .NET**
CDPCheckExtensions	Specifies whether SSL or TLS channels on this queue manager try to check CDP servers that are named in CrlDistributionPoint certificate extensions.	Yes	No	No	No	No
CertificateLabel	The certificate label of the channel definition.	Yes	No	No	No	No
CertificateValPolicy	Determines the type of certificate validation used.	Yes	No	No	No	No
ClientRevocationChecks	Determines how certificate revocation checking is configured if the client connect call uses an SSL/TLS channel.	Yes	No	No	No	No
EncryptionPolicySuiteB	Determines whether a channel uses Suite-B compliant cryptography and what level of strength is to be used.	Yes	No	No	No	No

mqclient.ini stanza name and attributes	Description	C and unmanaged .NET	Java	JMS	Managed .NET	Managed XMS .NET
OCSPAuthentication	Defines the behavior of IBM MQ when OCSP is enabled and the OCSP revocation check is unable to determine the certificate revocation status.	Yes	No	No	No	No
OCSPCheckExtensions	Controls whether IBM MQ acts on AuthorityInfoAccess certificate extensions.	Yes	No	No	No	No
SSLCryptoHardware	Sets the parameter string required to configure PKCS #11 cryptographic hardware present on the system.	Yes	No	No	No	No
SSLFipsRequired	Specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in IBM MQ.	Yes	No	No	No	No

Table 9. Which attributes apply to each type of client (continued)

Table 9. Which attributes apply to each type of client (continued)						
mqclient.ini stanza name and attributes	**Description**	**C and unmanaged .NET**	**Java**	**JMS**	**Managed .NET**	**Managed XMS .NET**
SSLHTTPProxyName	The string is either the host name or network address of the HTTP Proxy server that is to be used by GSKit for OCSP checks.	Yes	No	No	No	No
SSLKeyRepository	The location of the key repository that holds the user's digital certificate, in stem format.	Yes	No	No	No	No
SSLKeyResetCount	The number of unencrypted bytes sent and received on an SSL or TLS channel before the secret key is renegotiated.	Yes	No	No	No	No
TCP stanza						
ClntRcvBuffSize	The size in bytes of the TCP/IP receive buffer used by the client end of a client-connection server-connection channel.	Yes	Yes	Yes	Yes	Yes

Table 9. Which attributes apply to each type of client (continued)

mqclient.ini stanza name and attributes	Description	C and unmanaged .NET	Java	JMS	Managed .NET	Managed XMS .NET
ClntSndBuffSize	The size in bytes of the TCP/IP send buffer used by the client end of a client-connection server-connection channel.	Yes	Yes	Yes	Yes	Yes
Connect_Timeout	The number of seconds before an attempt to connect the socket times out.	Yes	Yes	Yes	No	No
IPAddressVersion	Specifies which IP protocol to use for a channel connection.	Yes	No	No	Yes	Yes
KeepAlive	Switches the KeepAlive function on or off.	Yes	Yes	Yes	Yes	Yes
Windows Library1	On Windows only, the name of the TCP/IP sockets DLL.	Yes	No	No	No	No

For the IBM MQ client for HP Integrity NonStop Server, you can use the TMF and TmfGateway stanzas to communicate with the TMF/Gateway.

CHANNELS stanza of the client configuration file

Use the CHANNELS stanza to specify information about client channels.

Note: The description of each attribute of this stanza indicates which IBM MQ clients can read that attribute. For a summary table for all IBM MQ MQI client configuration file stanzas, see Which IBM MQ attributes can be read by each client.

The following attributes can be included in the CHANNELS stanza.

`CCSID` **= *number***

The coded character set number to be used.

This attribute can be read by C, unmanaged .NET, managed .NET, and managed XMS .NET clients.

The CCSID number is equivalent to the MQCCSID environment parameter.

`ChannelDefinitionDirectory` = *path*

The directory path to the file containing the client channel definition table.

This attribute can be read by C, unmanaged .NET, managed .NET, and managed XMS .NET clients.

Windows On Windows systems, the default is the IBM MQ data and log files directory, typically `C:\ProgramData\IBM\MQ`.

Linux UNIX On UNIX and Linux systems, the default is `/var/mqm`.

The ChannelDefinitionDirectory path is equivalent to the MQCHLLIB environment parameter.

`ChannelDefinitionFile` = *filename* | `AMQCLCHL.TAB`

The name of the file containing the client channel definition table.

This attribute can be read by C, unmanaged .NET, managed .NET, and managed XMS .NET clients.

The client channel definition table is equivalent to the MQCHLTAB environment parameter.

`ReconDelay` = `(delay[,rand]) (delay[,rand])...`

The ReconDelay attribute provides an administrative option to configure reconnect delay for client programs that can auto-reconnect.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for JMS, managed .NET, and managed XMS .NET clients.

Here is an example configuration:

```
ReconDelay=(1000,200)(2000,200)(4000,1000)
```

The example shown defines an initial delay of one second, plus a random interval of up to 200 milliseconds. The next delay is two seconds plus a random interval of up to 200 milliseconds. All subsequent delays are four seconds, plus a random interval of up to 1000 milliseconds.

`DefRecon` = NO | YES | QMGR | DISABLED

The DefRecon attribute provides an administrative option to enable client programs to automatically reconnect, or to disable the automatic reconnection of a client program that has been written to reconnect automatically. You might opt to set the latter if a program uses an option, such as `MQPMO_LOGICAL_ORDER`, that is incompatible with reconnection.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for JMS, managed .NET, and managed XMS .NET clients.

Automatic client reconnection is not supported by IBM MQ classes for Java.

The interpretation of the DefRecon options depends on whether an `MQCNO_RECONNECT_*` value is also set in the client program, and what value is set.

If the client program connects using `MQCONN`, or sets the `MQCNO_RECONNECT_AS_DEF` option using `MQCONNX`, the reconnect value set by DefRecon takes effect. If no reconnect value is set in the program, or by the DefRecon option, the client program is not reconnected automatically.

NO

Unless overridden by `MQCONNX`, the client is not reconnected automatically.

YES

Unless overridden by `MQCONNX`, the client reconnects automatically.

QMGR

Unless overridden by `MQCONNX`, the client reconnects automatically, but only to the same queue manager. The QMGR option has the same effect as `MQCNO_RECONNECT_Q_MGR`.

DISABLED

Reconnection is disabled, even if requested by the client program using the `MQCONNX` MQI call.

The automatic client reconnection depends on two values:

- The reconnect option set in the application
- **DefRecon** value in the `mqclient.ini` file

Table 10. Automatic reconnection depends on the values set in the application and in the mqclient.ini file

DefRecon value in the `mqclient.ini`	**Reconnection options set in the application**			
	MQCNO_RECONNECT	MQCNO_RECONNECT_Q_MGR	MQCNO_RECONNECT_AS_DEF	MQCNO_RECONNECT_DISABLED
NO	YES	QMGR	NO	NO
YES	YES	QMGR	YES	NO
QMGR	YES	QMGR	QMGR	NO
DISABLED	NO	NO	NO	NO

MQReconnectTimeout

The timeout in seconds to reconnect to a client. The default value is 1800 seconds (30 minutes).

This attribute can be read by C and unmanaged .NET clients, and managed .NET clients.

IBM MQ classes for JMS clients can specify a timeout to reconnect using the connection factory property CLIENTRECONNECTTIMEOUT. The default value for this property is 1800 seconds (30 minutes).

IBM MQ classes for XMS .NET clients can specify a timeout to reconnect using the following properties:

- The connection factory property CLIENTRECONNECTTIMEOUT. The default value for this property is 1800 seconds (30 minutes). This property is valid only for Managed mode.
- The property XMSC.WMQ_CLIENT_RECONNECT_TIMEOUT. The default value for this property is 1800 seconds (30 minutes). This property is valid only for Managed mode.

ServerConnectionParms

ServerConnectionParms is equivalent to the MQSERVER environment parameter and specifies the location of the IBM MQ server and the communication method to be used.

This attribute can be read by C, unmanaged .NET, managed .NET, and managed XMS .NET clients.

The ServerConnectionParms attribute defines only a simple channel. You cannot use it to define an SSL channel or a channel with channel exits. It is a string of the format *ChannelName*/*TransportType*/*ConnectionName*, *ConnectionName* must be a fully qualified network name. *ChannelName* cannot contain the forward slash (/) character because this character is used to separate the channel name, transport type, and connection name.

When ServerConnectionParms is used to define a client channel, a maximum message length of 100 MB is used. Therefore the maximum message size in effect for the channel is the value specified in the SVRCONN channel on the server.

Note that only a single client channel connection can be made. For example, if you have two entries:

```
ServerConnectionParms=R1.SVRCONN/TCP/localhost(1963)
ServerConnectionParms=R2.SVRCONN/TCP/localhost(1863)
```

only the second one is used.

Specify *ConnectionName* as a comma-separated list of names for the stated transport type. Generally, only one name is required. You can provide multiple *hostnames* to configure multiple connections with the same properties. The connections are tried in the order that they are specified in the connection list until a connection is successfully established. If no connection is successful, the client starts to process again. Connection lists are an alternative to queue manager groups to configure connections for reconnectable clients.

Put1DefaultAlwaysSync = NO | YES
Controls the behavior of the MQPUT1 function call with the option MQPMO_RESPONSE_AS_Q_DEF.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for Java, and IBM MQ classes for JMS, managed .NET, and managed XMS .NET clients.

NO
If MQPUT1 is set with MQPMO_SYNCPOINT, it behaves as MQPMO_ASYNC_RESPONSE. Similarly, if MQPUT1 is set with MQPMO_NO_SYNCPOINT, it behaves as MQPMO_SYNC_RESPONSE. This is the default value.

YES
MQPUT1 behaves as if MQPMO_SYNC_RESPONSE is set, regardless of whether MQPMO_SYNCPOINT or MQPMO_NO_SYNCPOINT is set.

PasswordProtection = Compatible | always | optional
From IBM MQ 8.0, allows you to set protected passwords in the MQCSP structure, rather than using SSL or TLS.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for Java, and IBM MQ classes for JMS, managed .NET, and managed XMS .NET clients.

MQCSP password protection is useful for test and development purposes as using MQCSP password protection is simpler than setting up SSL/TLS encryption, but not as secure. For more information, see MQCSP password protection.

Related tasks

Connecting IBM MQ MQI applications to queue managers

ClientExitPath stanza of the client configuration file

Use the ClientExitPath stanza to specify the default locations of channel exits on the client.

Note: The description of each attribute of this stanza indicates which IBM MQ clients can read that attribute. For a summary table for all IBM MQ MQI client configuration file stanzas, see Which IBM MQ attributes can be read by each client.

The following attributes can be included in the ClientExitPath stanza.

ExitsDefaultPath = *string*
Specifies the location of 32-bit channel exits for clients.

This attribute can be read by C, unmanaged .NET, managed .NET, managed XMS .NET, IBM MQ classes for Java, and IBM MQ classes for JMS clients. IBM MQ classes for Java and IBM MQ classes for JMS clients use this attribute to locate 32-bit channel exits that are not written in Java.

ExitsDefaultPath64 = *string*
Specifies the location of 64-bit channel exits for clients.

This attribute can be read by C, unmanaged .NET, managed .NET, managed XMS .NET, IBM MQ classes for Java, and IBM MQ classes for JMS clients. IBM MQ classes for Java and IBM MQ classes for JMS clients use this attribute to locate 64-bit channel exits that are not written in Java.

JavaExitsClassPath = *string*
The values to be added to the classpath when a Java exit is run. This is ignored by exits in any other language.

This attribute can be read by IBM MQ classes for Java and IBM MQ classes for JMS clients.

In the JMS configuration file, the JavaExitsClassPath name is given the standard com.ibm.mq.cfg. prefix and this full name is also used on the IBM WebSphere MQ 7.0 or later system property.
At Version 6.0 this attribute was specified using system property com.ibm.mq.exitClasspath, which

was documented in the Version 6.0 readme. The use of com.ibm.mq.exitClasspath is deprecated. If both JavaExitsClassPath and exitClasspath are present, JavaExitsClassPath is honored. If only exitClasspath usage is present, it is still honored in IBM WebSphere MQ 7.0 or later.

JMQI stanza of the client configuration file

Use the JMQI stanza to specify configuration parameters for the Java Message Queuing Interface (JMQI) used by the IBM MQ classes for Java and IBM MQ classes for JMS.

Note: The description of each attribute of this stanza indicates which IBM MQ clients can read that attribute. For a summary table for all IBM MQ MQI client configuration file stanzas, see Which IBM MQ attributes can be read by each client.

The following attribute can be included in the JMQI stanza:

useMQCSPauthentication = NO | YES
: Controls whether IBM MQ classes for Java and IBM MQ classes for JMS applications should use Compatibility mode or MQCSP authentication mode when authenticating with a queue manager.

  This attribute can be read by IBM MQ classes for Java, and IBM MQ classes for JMS clients.

  This attribute can have the following values:

  NO
  : Use compatibility mode when authenticating with a queue manager. This is the default value.

  YES
  : Use MQCSP authentication mode when authenticating with a queue manager.

  For more information about Compatibility mode and MQCSP authentication mode, see Connection authentication with the Java client.

Windows LU62, NETBIOS, and SPX stanzas of the client configuration file

On Windows systems only, use these stanzas to specify configuration parameters for the specified network protocols.

LU62 stanza

Use the LU62 stanza to specify SNA LU 6.2 protocol configuration parameters. The following attributes can be included in this stanza:

`Library1` = ***DLLName*** | **`WCPIC32`**
: The name of the APPC DLL.

`Library2` = ***DLLName*** | **`WCPIC32`**
: The same as Library1, used if the code is stored in two separate libraries. .

TPName
: The TP name to start on the remote site.

NETBIOS stanza

Use the NETBIOS stanza to specify NetBIOS protocol configuration parameters. The following attributes can be included in this stanza:

`AdapterNum` = ***number*** | **`0`**
: The number of the LAN adapter.

`Library1` = ***DLLName*** | **`NETAPI32`**
: The name of the NetBIOS DLL.

`LocalName` = ***name***
: The name by which this computer is known on the LAN.

  This is equivalent to the MQNAME environment parameter.

NumCmds = *number* | **1**
How many commands to allocate.

NumSess = *number* | **1**
How many sessions to allocate.

SPX stanza

Use the SPX stanza to specify SPX protocol configuration parameters. The following attributes can be included in this stanza:

BoardNum = *number* | **0**
The LAN adapter number.

KeepAlive = **YES** | **NO**
Switch the KeepAlive function on or off.

KeepAlive = YES causes SPX to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

Library1 = *DLLName* | **WSOCK32.DLL**
The name of the SPX DLL.

Library2 = *DLLName* | **WSOCK32.DLL**
The same as Library1, used if the code is stored in two separate libraries.

Socket = *number* | **5E86**
The SPX socket number in hexadecimal notation.

MessageBuffer stanza of the client configuration file

Use the MessageBuffer stanza to specify information about message buffers.

Note: The description of each attribute of this stanza indicates which IBM MQ clients can read that attribute. For a summary table for all IBM MQ MQI client configuration file stanzas, see Which IBM MQ attributes can be read by each client.

The following attributes can be included in the MessageBuffer stanza:

MaximumSize = *integer* | **1**
Size, in kilobytes, of the read-ahead buffer, in the range 1 - 999 999.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for Java, IBM MQ classes for JMS, managed .NET, and managed XMS .NET clients.

The following special values exist:

-1
The client determines the appropriate value.

0
Read ahead is disabled for the client.

PurgeTime = *integer* | **600**
Interval, in seconds, after which messages left in the read-ahead buffer are purged.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for Java, IBM MQ classes for JMS, managed .NET, and managed XMS .NET clients.

If the client application is selecting messages based on MsgId or CorrelId, it is possible that the read ahead buffer might contain messages that are sent to the client with a previously requested MsgId or CorrelId. These messages would then be stranded in the read ahead buffer until an MQGET is issued with an appropriate MsgId or CorrelId. You can purge messages from the read ahead buffer by setting PurgeTime. Any messages that have remained in the read ahead buffer for longer than the purge interval are automatically purged. These messages have already been removed from the queue on the queue manager, so unless they are being browsed, they are lost.

The valid range is in the range 1 - 999 999 seconds, or the special value 0, meaning that no purge takes place.

`UpdatePercentage` = *integer* | <u>`-1`</u>

The update percentage value, in the range of 1 - 100, used in calculating the threshold value to determine when a client application makes a new request to the server. The special value -1 indicates that the client determines the appropriate value.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for Java, IBM MQ classes for JMS, managed .NET, and managed XMS .NET clients.

The client periodically sends a request to the server indicating how much data the client application has consumed. A request is sent when the number of bytes, *n*, retrieved by the client by way of MQGET calls exceeds a threshold *T*. *n* is reset to zero each time that a new request is sent to the server.

The threshold *T* is calculated as follows:

```
T = Upper - Lower
```

Upper is the same as the read-ahead buffer size, specified by the MaximumSize attribute, in Kilobytes. Its default is 100 Kb.

Lower is lower than Upper, and is specified by the UpdatePercentage attribute. This attribute is a number in the range 1 - 100, and has a default of 20. Lower is calculated as follows:

```
Lower = Upper x UpdatePercentage / 100
```

Example 1:

The MaximumSize and UpdatePercentage attributes take their defaults of 100 Kb and 20 Kb.

The client calls MQGET to retrieve a message, and does so repeatedly. This continues until MQGET has consumed *n* bytes.

Using the calculation

```
T = Upper - Lower
```

T is (100 - 20) = 80 Kb.

So when MQGET calls have removed 80 Kb from a queue, the client makes a new request automatically.

Example 2:

The MaximumSize attribute takes its default of 100 Kb, and a value of 40 is chosen for UpdatePercentage.

The client calls MQGET to retrieve a message, and does so repeatedly. This continues until MQGET has consumed *n* bytes.

Using the calculation

```
T = Upper - Lower
```

T is (100 - 40) = 60 Kb

So when MQGET calls have removed 60 Kb from a queue, the client makes a new request automatically. This is sooner than in EXAMPLE 1 where the defaults were used.

Therefore, choosing a larger threshold *T* tends to decrease the frequency at which requests are sent from client to server. Conversely, choosing a smaller threshold *T* tends to increase the frequency of requests that are sent from client to server.

However, choosing a large threshold *T* can mean that the performance gain of read ahead is reduced as the chance of the read ahead buffer becoming empty can increase. When this happens an MQGET call might have to pause, waiting for data to arrive from the server.

PreConnect stanza of the client configuration file

Use the PreConnect stanza to configure the PreConnect exit in the `mqclient.ini` file.

Note: The description of each attribute of this stanza indicates which IBM MQ clients can read that attribute. For a summary table for all IBM MQ MQI client configuration file stanzas, see Which IBM MQ attributes can be read by each client.

The following attributes can be included in the PreConnect stanza:

`Data` = *<URL>*
URL of the repository where connection definitions are stored.

This attribute can be read by C and unmanaged .NET clients.

For example, when using an LDAP server:

```
Data = ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
```

`Function` = *<myFunc>*
Name of the functional entry point into the library that contains the PreConnect exit code.

This attribute can be read by C and unmanaged .NET clients.

The function definition adheres to the PreConnect exit prototype MQ_PRECONNECT_EXIT.

The maximum length of this field is MQ_EXIT_NAME_LENGTH.

`Module` = *<amqldapi>*
The name of the module containing the API exit code.

This attribute can be read by C and unmanaged .NET clients.

If this field contains the full path name of the module, it is used as is.

Sequence = *<sequence_number>*
The sequence in which this exit is called relative to other exits. An exit with a low sequence number is called before an exit with a higher sequence number. There is no need for the sequence numbering of exits to be continuous; a sequence of 1, 2, 3 has the same result as a sequence of 7, 42, 1096. This attribute is an unsigned numeric value.

This attribute can be read by C and unmanaged .NET clients.

Multiple PreConnect stanzas can be defined within the `mqclient.ini` file. The processing order of each exit is determined by the Sequence attribute of the stanza.

Related tasks

Referencing connection definitions using a pre-connect exit from a repository

Security stanza of the client configuration file

Use the Security stanza to disable or enable AMS for client connections to a queue manager.

Note: The description of each attribute of this stanza indicates which IBM MQ clients can read that attribute. For a summary table for all IBM MQ MQI client configuration file stanzas, see Which IBM MQ attributes can be read by each client.

The following attribute can be included in the Security stanza:

DisableClientAMS = NO|YES
The DisableClientAMS attribute allows you to disable IBM MQ Advanced Message Security (AMS) if you are using a Version 7.5 or later client to connect to a queue manager from an earlier version of the product and a `2085 (MQRC_UNKNOWN_OBJECT_NAME)` error is reported.

From Version 7.5, IBM MQ Advanced Message Security (AMS) is automatically enabled in an IBM MQ client and so, by default, the client tries to check the security policies for objects at the queue manager. However, servers on earlier versions of the product, for example Version 7.1, do not have AMS enabled and this causes 2085 (MQRC_UNKNOWN_OBJECT_NAME) error to be reported.

The following examples show how to use the DisableClientAMS attribute:

- To disable AMS:

```
Security:
DisableClientAMS=Yes
```

- To enable AMS:

```
Security:
DisableClientAMS=No
```

This attribute can be read by C, IBM MQ classes for Java, and IBM MQ classes for JMS clients.

Related tasks

Disabling Advanced Message Security at the client

SSL stanza of the client configuration file

Use the SSL stanza to specify information about the use of SSL or TLS.

Note: The description of each attribute of this stanza indicates which IBM MQ clients can read that attribute. For a summary table for all IBM MQ MQI client configuration file stanzas, see Which IBM MQ attributes can be read by each client.

The following attributes can be included in the SSL stanza:

CDPCheckExtensions= YES | NO

CDPCheckExtensions specifies whether SSL or TLS channels on this queue manager try to check CDP servers that are named in CrlDistributionPoint certificate extensions.

This attribute can be read by C and unmanaged .NET clients.

This attribute has the following possible values:

- YES: SSL or TLS channels try to check CDP servers to determine whether a digital certificate is revoked.
- NO: SSL or TLS channels do not try to check CDP servers. This value is the default.

CertificateLabel = *string*

The certificate label of the channel definition.

This attribute can be read by C and unmanaged .NET clients.

For more information, see Certificate label (CERTLABL).

CertificateValPolicy = *string*

Determines the type of certificate validation used.

This attribute can be read by C and unmanaged .NET clients.

This attribute has the following possible values:

ANY

Use any certificate validation policy supported by the underlying secure sockets library. This setting is the default setting.

RFC5280

Use only certificate validation which complies with the RFC 5280 standard.

ClientRevocationChecks = REQUIRED | OPTIONAL | DISABLED

Determines how certificate revocation checking is configured if the client connect call uses an SSL/TLS channel. See also **OCSPAuthentication**.

This attribute can be read by C and unmanaged .NET clients.

This attribute has the following possible values:

REQUIRED (default)
Attempts to load certificate revocation configuration from the CCDT and perform revocation checking as configured. If the CCDT file cannot be opened or it is not possible to validate the certificate (because an OCSP or CRL server is not available, for example) the MQCONN call fails. No revocation checking is performed if the CCDT contains no revocation configuration but this does not cause the channel to fail.

Windows On Windows systems, you can also use Active Directory for CRL revocation checking. You cannot use Active Directory for OCSP revocation checking.

OPTIONAL
As for `REQUIRED`, but if it is not possible to load the certificate revocation configuration, the channel does not fail.

DISABLED
No attempt is made to load certificate revocation configuration from the CCDT and no certificate revocation checking is done.

Note: If you are using MQCONNX rather than MQCONN calls, you might choose to supply authentication information records (MQAIR) via the MQSCO. The default behavior with MQCONNX is therefore not to fail if the CCDT file cannot be opened but to assume that you are supplying an MQAIR (even if you choose not to do so).

`EncryptionPolicySuiteB` = *string*
Determines whether a channel uses Suite-B compliant cryptography and what level of strength is to be used.

This attribute can be read by C and unmanaged .NET clients.

This attribute has the following possible values:

NONE
Suite-B compliant cryptography is not used. This setting is the default setting.

128_BIT,192_BIT
Sets the security strength to both 128-bit and 192-bit levels.

128_BIT
Sets the security strength to 128-bit level.

192_BIT
Sets the security strength to 192-bit level.

`OCSPAuthentication` = `OPTIONAL` | `REQUIRED` | `WARN`
Defines the behavior of IBM MQ when OCSP is enabled and the OCSP revocation check is unable to determine the certificate revocation status. See also **`ClientRevocationChecks`**.

This attribute can be read by C and unmanaged .NET clients.

This attribute has the following possible values:

OPTIONAL
Any certificate with a revocation status that cannot be determined by OCSP checking is accepted and no warning or error message is generated. The SSL or TLS connection continues as if no revocation check had been made.

REQUIRED
OCSP checking must yield a definitive revocation result for every SSL or TLS certificate which is checked. Any SSL or TLS certificate with a revocation status that cannot be verified is rejected with an error message. If queue manager SSL event messages are enabled, an MQRC_CHANNEL_SSL_ERROR message with a ReasonQualifier of MQRQ_SSL_HANDSHAKE_ERROR is generated. The connection is closed.

This value is the default value.

WARN
A warning is reported in the queue manager error logs if an OCSP revocation check is unable to determine the revocation status of any SSL or TLS certificate. If queue manager SSL event messages are enabled, an MQRC_CHANNEL_SSL_WARNING message with a ReasonQualifier of MQRQ_SSL_UNKNOWN_REVOCATION is generated. The connection is allowed to continue.

`OCSPCheckExtensions` = <u>YES</u> | `NO`
Controls whether IBM MQ acts on AuthorityInfoAccess certificate extensions.

This attribute can be read by C and unmanaged .NET clients.

If the value is set to `NO`, IBM MQ ignores AuthorityInfoAccess certificate extensions and does not attempt an OCSP security check. The default value is YES.

`SSLCryptoHardware` = *string*
Sets the parameter string required to configure PKCS #11 cryptographic hardware present on the system.

This attribute can be read by C and unmanaged .NET clients.

Specify a string in the following format: `GSK_PKCS11` = *driver path and filename* `;` *token label* `;` *token password* `;` *symmetric cipher setting* ;

For example: `GSK_PKCS11=/usr/lib/pkcs11/ PKCS11_API.so;tokenlabel;passw0rd;SYMMETRIC_CIPHER_ON`

The driver path is an absolute path to the shared library providing support for the PKCS #11 card. The driver file name is the name of the shared library. An example of the value required for the PKCS #11 driver path and file name is `/usr/lib/pkcs11/PKCS11_API.so`. To access symmetric cipher operations through GSKit, specify the symmetric cipher setting parameter. This parameter can have either of the following values:

SYMMETRIC_CIPHER_OFF
Do not access symmetric cipher operations. This setting is the default setting.

SYMMETRIC_CIPHER_ON
Access symmetric cipher operations.

The maximum length of the string is 256 characters. The default value is blank. If you specify a string that is not in the correct format, an error is generated.

`SSLFipsRequired` = YES | <u>NO</u>
Specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in IBM MQ.

This attribute can be read by C, and unmanaged .NET clients.

If cryptographic hardware is configured, the cryptographic modules used are those modules provided by the hardware product. These might, or might not, be FIPS-certified to a particular level, depending on the hardware product in use.

`SSLHTTPProxyName` = *string*
The string is either the host name or network address of the HTTP Proxy server that is to be used by GSKit for OCSP checks. This address can be followed by an optional port number, enclosed in parentheses. If you do not specify the port number, the default HTTP port, 80, is used.

This attribute can be read by C, and unmanaged .NET clients.

On the HP-UX PA-RISC and Sun Solaris SPARC platforms, and for 32-bit clients on AIX, the network address can be only an IPv4 address; on other platforms it can be an IPv4 or IPv6 address.

This attribute might be necessary if, for example, a firewall prevents access to the URL of the OCSP responder.

`SSLKeyRepository` = *pathname*
The location of the key repository that holds the user's digital certificate, in stem format. That is, it includes the full path and the file name without an extension.

This attribute can be read by C, and unmanaged .NET clients.

`SSLKeyResetCount` = *integer* | 0
The number of unencrypted bytes sent and received on an SSL or TLS channel before the secret key is renegotiated.

This attribute can be read by C, and unmanaged .NET clients.

The value must be in the range 0 - 999999999.

The default is 0, which means that secret keys are never renegotiated.

If you specify a value of 1 - 32768, SSL or TLS channels use a secret key reset count of 32768 (32Kb). This is to avoid excessive key resets, which would occur for small secret key reset values.

TCP stanza of the client configuration file

Use the TCP stanza to specify TCP network protocol configuration parameters.

Note: The description of each attribute of this stanza indicates which IBM MQ clients can read that attribute. For a summary table for all IBM MQ MQI client configuration file stanzas, see Which IBM MQ attributes can be read by each client.

The following attributes can be included in the TCP stanza:

`ClntRcvBuffSize` = *number* | 0
The size in bytes of the TCP/IP receive buffer used by the client end of a client-connection server-connection channel.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for Java, IBM MQ classes for JMS, managed .NET, and managed XMS .NET clients.

A value of zero indicates that the operating system will manage the buffer sizes, as opposed to the buffer sizes being fixed by IBM MQ. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

`ClntSndBuffSize` = *number* | 0
The size in bytes of the TCP/IP send buffer used by the client end of a client-connection server-connection channel.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for Java, IBM MQ classes for JMS, managed .NET, and managed XMS .NET clients.

A value of zero indicates that the operating system will manage the buffer sizes, as opposed to the buffer sizes being fixed by IBM MQ. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

`Connect_Timeout` = *number*
The number of seconds before an attempt to connect the socket times out. The default value of zero specifies that there is no connect timeout.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for Java, and IBM MQ classes for JMS clients.

IBM MQ channel processes connect over nonblocking sockets. Therefore, if the other end of the socket is not ready, connect() returns immediately with *EINPROGRESS* or *EWOULDBLOCK* . Following this, connect will be attempted again, up to a total of 20 such attempts, when a communications error is reported.

If Connect_Timeout is set to a non-zero value, IBM MQ waits for the stipulated period over select() call for the socket to get ready. This increases the chances of success of a subsequent connect() call. This option might be beneficial in situations where connects would require some waiting period, due to high load on the network.

There is no relationship between the Connect_Timeout, ClntSndBuffSize, and ClntRcvBuffSize parameters.

`IPAddressVersion` = MQIPADDR_IPV4 | MQIPADDR_IPV6

Specifies which IP protocol to use for a channel connection.

This attribute can be read by C, unmanaged .NET, managed .NET, and managed XMS .NET clients.

It has the possible string values of `MQIPADDR_IPV4` or `MQIPADDR_IPV6`. These values have the same meanings as `IPV4` and `IPV6` in **`ALTER QMGR IPADDRV`**.

`KeepAlive` = YES | NO

Switch the KeepAlive function on or off. KeepAlive=YES causes TCP/IP to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

This attribute can be read by C, unmanaged .NET, IBM MQ classes for Java, IBM MQ classes for JMS, managed .NET, and managed XMS .NET clients.

Windows **`Library1` = *DLLName* | WSOCK32**

(Windows only) The name of the TCP/IP sockets DLL.

This attribute can be read by C and unmanaged .NET clients.

TMF and TmfGateway stanzas

Use the TMF and TMF/Gateway stanzas to specify the required configuration parameters for the IBM MQ client for HP Integrity NonStop Server to communicate with the TMF/Gateway.

If you want to use the HP NonStop Transaction Management Facility (TMF), then you must define a TMF stanza and one TmfGateway stanza for each queue manager with which you are communicating. All values are derived from your configuration.

The IBM MQ provided TMF/Gateway runs in a Pathway environment.

TMF stanza

`PathMon` = *name*

The name of your defined Pathmon process that defines the server classes for the TMF/Gateway.

TmfGateway stanza

The following attributes can be included in this stanza:

`QManager` = *name*

The name of the queue manager.

`Server` = *name*

The server class name for the TMF/Gateway configured for that queue manager.

Example

Here is an example of a TMF stanza that is defined with two TmfGateway stanzas for two different queue managers on different servers:

```
TMF:
PathMon=$PSD1P

TmfGateway:
QManager=MQ5B
Server=MQ-MQ5B

TmfGateway:
QManager=MQ5C
Server=MQ-MQ5C
```

Related concepts

"Gateway process overview" on page 515

The HP NonStop Transaction Management Facility (TMF) provides services to enable a gateway process to register as a resource manager. The IBM WebSphere MQ for HP Integrity NonStop Server provided TMF/Gateway process runs under Pathway.

“Configuring the client initialization file” on page 517
If you are using the HP NonStop Transaction Management Facility (TMF), you must have an IBM MQ client initialization file to enable your IBM MQ client for the HP Integrity NonStop Server to reach the TMF Gateway.

Using IBM MQ environment variables

This section describes the environment variables that you can use with IBM MQ MQI client applications.

You can use environment variables in the following ways:

- Set the variables in your system profile to make a permanent change
- Issue a command from the command line to make a change for this session only
- To give one or more variables a particular value dependent on the application that is running, add commands to a command script file used by the application

IBM MQ uses default values for those variables that you have not set.

Commands are available on all the IBM MQ MQI client platforms unless otherwise stated.

Note: z/OS IBM MQ for z/OS does not support any IBM MQ environment variables. If you are using this platform as your server, see Client channel definition table for information about how the client channel definition table is generated on z/OS. You can still use the IBM MQ environment variables on your client platform.

For each environment variable, use the command relevant to your platform to display the current setting or to reset the value of a variable.

On Windows, use the following commands:

- To remove the value of an environment variable, use the command `SET MQSERVER=`
- To display the current setting of an environment variable, use the command `SET MQSERVER`
- To display all environment variables for the session, use the command `set`

On UNIX and Linux systems, use the following commands:

- To remove the value of an environment variable, use the command `unset MQSERVER`
- To display the current setting of an environment variable, use the command `echo $MQSERVER`
- To display all environment variables for the session, use the command `set`

For information about the individual variables, see the following subtopics:

Related concepts

“Configuring a client using a configuration file” on page 47
Configure your clients by using attributes in a text file. These attributes can be overridden by environment variables or in other platform-specific ways.

Related reference

Environment variables

MQCCSID

MQCCSID specifies the coded character set number to be used and overrides the CCSID value with which the server has been configured.

See Choosing client or server coded character set identifier (CCSID) for more information.

To set this variable use one of these commands:

- For Windows:

```
SET MQCCSID=number
```

- For UNIX and Linux systems:

```
export MQCCSID=number
```

- IBM i For IBM i:

```
ADDENVVAR ENVVAR(MQCCSID) VALUE(number)
```

MQCERTLABL

MQCERTLABL specifies the certificate label of the channel definition.

See Certificate label (CERTLABL) for more information.

MQCERTVPOL

MQCERTVPOL specifies the certificate validation policy used.

For more information about certificate validation policies in IBM MQ, see Certificate validation policies in IBM MQ.

This environment variable overrides the *CertificateValPolicy* setting in the SSL stanza of the client ini file. The variable can be set to one of two values:

ANY
: Use any certificate validation policy supported by the underlying secure sockets library.

RFC5280
: Use only certificate validation which complies with the RFC 5280 standard.

To set this variable, use one of these commands:

- For Windows:

```
SET MQCERTVPOL= value
```

- For UNIX and Linux systems:

```
export MQCERTVPOL= value
```

- IBM i For IBM i:

```
ADDENVVAR ENVVAR(MQCERTVPOL) VALUE(value)
```

MQCHLLIB

`MQCHLLIB` specifies the directory path to the file containing the client channel definition table (CCDT). The file is created on the server, but can be copied across to the IBM MQ MQI client workstation.

If `MQCHLLIB` is not set, the path for the client defaults to:

- Windows For Windows: *MQ_INSTALLATION_PATH*
- Linux UNIX For UNIX and Linux systems: `/var/mqm/`
- IBM i For IBM i: `/QIBM/UserData/mqm/`

For the **crtmqm** and **strmqm** commands, the path defaults to one of two sets of paths. If *datapath* is set, the path defaults to one of the first set. If *datapath* is not set, the path defaults to one of the second set.

- Windows For Windows: `datapath\@ipcc`
- Linux UNIX For UNIX and Linux systems: `datapath/@ipcc`
- IBM i For IBM i: `datapath/&ipcc`

Or:

- Windows For Windows: `MQ_INSTALLATION_PATH\data\qmgrs\qmgrname\@ipcc`
- Linux UNIX For UNIX and Linux systems: `/prefix/qmgrs/qmgrname/@ipcc`
- IBM i For IBM i: `/prefix/qmgrs/qmgrname/&ipcc`

where:

- *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.
- If present, *datapath* is the value of DataPath defined in the queue manager stanza.
- *prefix* is the value of Prefix defined in the queue manager stanza. Prefix is typically /var/mqm on UNIX and Linux platforms IBM i, and /QIBM/UserData/mqm/ on IBM i.
- *qmgrname* is the value of the Directory attribute defined in the queue manager stanza. The value might be different from the actual queue manager name. The value might have been altered to replace special characters.
- The queue manager stanza is defined in the mqs.ini file on IBM i IBM, UNIX, and Linux, and in the registry on Windows

Notes:

1. z/OS If you are using IBM MQ for z/OS as your server, the file must be kept on the IBM MQ client workstation.
2. If set, MQCHLLIB overrides the path used to locate the CCDT.
3. Environment variables, such as MQCHLLIB, can be scoped to a process, or a job, or system-wide, in a platform-specific way.
4. If you set MQCHLLIB system-wide on a server, it sets the same path to the CCDT file for all the queue managers on the server. If you do not set the MQCHLLIB environment variable, the path is different for each queue manager. Queue managers read the value of MQCHLLIB, if it is set, on either the **crtmqm** or **strmqm** command.
5. If you create multiple queue managers on one server, the distinction is important, for the following reason. If you set MQCHLLIB system-wide, each queue manager updates the same CCDT file. The file contains the client-connection definitions from all the queue managers on the server. If the same definition exists on multiple queue managers, SYSTEM.DEF.CLNTCONN for example, the file contains the latest definition. When you create a queue manager, if MQCHLLIB is set, SYSTEM.DEF.CLNTCONN is updated in the CCDT. The update overwrites the SYSTEM.DEF.CLNTCONN created by a different queue manager. If you modified the earlier definition, your modifications are lost. For this reason, you must consider finding alternatives to setting MQCHLLIB as a system-wide environment variable on the server.
6. The MQSC and PCF NOREPLACE option on a client-connection definition does not check the contents of the CCDT file. A client-connection channel definition of the same name that was previously created, but not by this queue manager, is replaced, regardless of the NOREPLACE option. If the definition was previously created by the same queue manager, the definition is not replaced.
7. The command, **rcrmqobj** -t clchltab deletes and recreates the CCDT file. The file is recreated with only the client-connection definitions created on the queue manager that the command is running against.

8. Other commands that update the CCDT modify only the client-connection channels that have the same channel name. Other client-connection channels in the file are not altered.
9. The path for MQCHLLIB does not need quotations marks.

Examples

To set this variable use one of these commands:

- Windows For Windows:

  ```
  SET MQCHLLIB=pathname
  ```

  For example:

  ```
  SET MQCHLLIB=C:\wmqtest
  ```

- Linux UNIX For UNIX and Linux systems:

  ```
  export MQCHLLIB=pathname
  ```

- IBM i For IBM i:

  ```
  ADDENVVAR ENVVAR(MQCHLLIB) VALUE(pathname)
  ```

MQCHLTAB

MQCHLTAB specifies the name of the file containing the client channel definition table (ccdt). The default file name is AMQCLCHL.TAB.

For information about where the client channel definition table is located on a server, see "Client channel definition table" on page 38.

To set this variable use one of these commands:

- On Windows:

  ```
  SET MQCHLTAB=filename
  ```

- On UNIX and Linux systems:

  ```
  export MQCHLTAB=filename
  ```

- IBM i On IBM i:

  ```
  ADDENVVAR ENVVAR(MQCHLTAB) VALUE(filename)
  ```

For example:

```
SET MQCHLTAB=ccdf1.tab
```

In the same way as for the client, the MQCHLTAB environment variable on the server specifies the name of the client channel definition table.

MQIPADDRV

MQIPADDRV specifies which IP protocol to use for a channel connection. It has the possible string values of "MQIPADDR_IPv4" or "MQIPADDR_IPv6". These values have the same meanings as IPv4 and IPv6 in ALTER QMGR IPADDRV. If it is not set, "MQIPADDR_IPv4" is assumed.

To set this variable use one of these commands:

- For Windows:

```
SET MQIPADDRV=MQIPADDR_IPv4|MQIPADDR_IPv6
```

- For UNIX and Linux systems:

```
export MQIPADDRV=MQIPADDR_IPv4|MQIPADDR_IPv6
```

- IBM i For IBM i:

```
ADDENVVAR ENVVAR(MQIPADDRV) VALUE(MQIPADDR_IPv4|MQIPADDR_IPv6)
```

MQNAME

MQNAME specifies the local NetBIOS name that the IBM MQ processes can use.

See "Defining a NetBIOS connection on Windows" on page 187 for a full description and for the rules of precedence on the client and the server.

To set this variable use this command:

```
SET MQNAME=Your_env_Name
```

For example:

```
SET MQNAME=CLIENT1
```

The NetBIOS on some platforms requires a different name (set by MQNAME) for each application if you are running multiple IBM MQ applications simultaneously on the IBM MQ MQI client.

MQSERVER

MQSERVER environment variable is used to define a minimal channel. MQSERVER specifies the location of the IBM MQ server and the communication method to be used.

You cannot use MQSERVER to define an SSL channel or a channel with channel exits. For details of how to define an SSL channel, see Protecting channels with SSL.

ConnectionName must be a fully-qualified network name. The *ChannelName* cannot contain the forward slash (/) character because this character is used to separate the channel name, transport type, and connection name. When the MQSERVER environment variable is used to define a client channel, a maximum message length (MAXMSGL) of 100 MB is used. Therefore the maximum message size in effect for the channel is the value specified in the SVRCONN channel at the server.

To set this variable use one of these commands:

- For Windows:

```
SET  MQSERVER=ChannelName/TransportType/ConnectionName
```

- For UNIX and Linux systems:

```
export  MQSERVER='ChannelName/TransportType/ConnectionName'
```

- IBM i For IBM i:

```
ADDENVVAR ENVVAR(MQSERVER) VALUE('ChannelName/TransportType/ConnectionName')
```

- z/OS For z/OS

```
export MQSERVER='SYSTEM.DEF.SVRCONN/TCP/AMACHINE.ACOMPANY.COM(1414) '
```

TransportType can be one of the following values, depending on your IBM MQ client platform:

- LU62
- TCP
- NETBIOS
- SPX

ConnectionName can be a comma-separated list of connection names. The connection names in the list are used in a similar way to multiple connections in a client connection table. The *ConnectionName* list might be used as an alternative to queue manager groups to specify multiple connections for the client to try. If you are configuring a multi-instance queue manager, you might use a *ConnectionName* list to specify different queue manager instances.

TCP/IP default port

By default, for TCP/IP, IBM MQ assumes that the channel will be connected to port 1414.

You can change this by:

- Adding the port number in brackets as the last part of the ConnectionName:
  - For Windows:

    ```
    SET  MQSERVER=ChannelName/TransportType/ConnectionName(PortNumber)
    ```

  - For UNIX and Linux systems:

    ```
    export  MQSERVER='ChannelName/TransportType/ConnectionName(PortNumber)'
    ```

- Changing the `mqclient.ini` file by adding the port number to the protocol name, for example:

```
TCP:
port=2001
```

- Adding IBM MQ to the services file as described in “Using the TCP/IP listener” on page 191.

SPX default socket

By default, for SPX, IBM MQ assumes that the channel will be connected to socket 5E86.

You can change this by:

- Adding the socket number in brackets as the last part of the ConnectionName:

```
SET  MQSERVER=ChannelName/TransportType/ConnectionName(SocketNumber)
```

For SPX connections, specify the ConnectionName and socket in the form `network.node(socket)`. If the IBM MQ client and server are on the same network, the network need not be specified. If you are using the default socket, the socket need not be specified.

- Changing the `qm.ini` file by adding the port number to the protocol name, for example:

```
SPX:
socket=5E87
```

Using MQSERVER

If you use the MQSERVER environment variable to define the channel between your IBM MQ MQI client machine and a server machine, this is the only channel available to your application, and no reference is made to the client channel definition table (CCDT).

In this situation, the listener program that you have running on the server machine determines the queue manager to which your application will connect. It will be the same queue manager as the listener program is connected to.

If the MQCONN or MQCONNX request specifies a queue manager other than the one the listener is connected to, or if the MQSERVER parameter *TransportType* is not recognized, the MQCONN or MQCONNX request fails with return code MQRC_Q_MGR_NAME_ERROR.

On UNIX and Linux systems, you might define MQSERVER as in one of the following examples:

```
export MQSERVER=CHANNEL1/TCP/'9.20.4.56(2002)'
export MQSERVER=CHANNEL1/LU62/BOX99
```

All `MQCONN` or `MQCONNX` requests then attempt to use the channel you have defined unless an MQCD structure has been referenced from the MQCNO structure supplied to `MQCONNX` , in which case the channel specified by the MQCD structure takes priority over any specified by the MQSERVER environment variable.

The MQSERVER environment variable takes priority over any client channel definition pointed to by MQCHLLIB and MQCHLTAB.

Canceling MQSERVER

To cancel MQSERVER and return to the client channel definition table pointed to by MQCHLLIB and MQCHLTAB, enter the following:

- On Windows:

```
SET MQSERVER=
```

- On UNIX and Linux systems:

```
unset MQSERVER
```

MQSSLCRYP

MQSSLCRYP holds a parameter string that allows you to configure the cryptographic hardware present on the system. The permitted values are the same as for the **SSLCRYP** parameter of the **`ALTER QMGR`** command.

To set this variable use one of these commands:

- On Windows systems:

```
SET MQSSLCRYP=string
```

- On UNIX and Linux systems:

```
export MQSSLCRYP=string
```

Related reference

ALTER QMGR command **SSLCRYP** parameter

MQSSLFIPS

MQSSLFIPS specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in IBM MQ. The values are the same as for the SSLFIPS parameter of the ALTER QMGR command.

The use of FIPS-certified algorithms is affected by the use of cryptographic hardware, see Specifying that only FIPS-certified CipherSpecs are used at run time on the MQI client.

To set this variable use one of these commands:

- On Windows systems:

```
SET MQSSLFIPS=YES|NO
```

- On UNIX and Linux systems:

```
export MQSSLFIPS=YES|NO
```

- IBM i On IBM i:

```
ADDENVVAR ENVVAR(MQSSLFIPS) VALUE(YES|NO)
```

The default is NO.

MQSSLKEYR

MQSSLKEYR specifies the location of the key repository that holds the digital certificate belonging to the user, in stem format. Stem format means that it includes the full path and the file name without an extension.

For full details, see the SSLKEYR parameter of the ALTER QMGR command.

To set this variable use one of these commands:

- On Windows systems:

```
SET MQSSLKEYR=pathname
```

- On UNIX and Linux systems:

```
export MQSSLKEYR=pathname
```

- IBM i On IBM i:

```
ADDENVVAR ENVVAR(MQSSLKEYR) VALUE(pathname)
```

There is no default value.

MQSSLPROXY

MQSSLPROXY specifies the host name and port number of the HTTP proxy server to be used by GSKit for OCSP checks.

To set this variable use one of these commands:

- On Windows systems:

```
SET MQSSLPROXY= string
```

- On UNIX and Linux systems:

```
export MQSSLPROXY="string"
```

The string is either the host name or network address of the HTTP Proxy server which is to be used by GSKit for OCSP checks. This address can be followed by an optional port number, enclosed in parentheses. If you do not specify the port number, the default HTTP port, 80, is used.

For example, on UNIX and Linux systems, you can use the one of the following commands:

-
```
export MQSSLPROXY="proxy.example.com(80)"
```

-
```
export MQSSLPROXY="127.0.0.1"
```

MQSSLRESET

MQSSLRESET represents the number of unencrypted bytes sent and received on an SSL or TLS channel before the secret key is renegotiated.

See Resetting SSL and TLS secret keys for more information about secret key renegotiation.

It can be set to an integer in the range 0 through 999 999 999. The default is 0, which indicates that secret keys are never renegotiated. If you specify an SSL or TLS secret key reset count in the range 1 byte through 32 KB, SSL or TLS channels use a secret key reset count of 32 KB. This secret reset count is to avoid excessive key resets which would occur for small SSL or TLS secret key reset values.

To set this variable use one of these commands:

- On Windows systems:

```
SET MQSSLRESET=integer
```

- On UNIX and Linux systems:

```
export MQSSLRESET=integer
```

- IBM i On IBM i:

```
ADDENVVAR ENVVAR(MQSSLRESET) VALUE(integer)
```

MQSUITEB

You can configure IBM MQ to operate in compliance with the NSA Suite B standard on Windows, UNIX and Linux platforms.

Suite B restricts the set of enabled cryptographic algorithms in order to provide an assured level of security.

See Configuring IBM MQ for Suite B for more information.

MQTCPTIMEOUT

How long IBM MQ waits for a TCP connect call.

Changing IBM MQ and queue manager configuration information

Change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

You can change IBM MQ configuration information by changing the values specified on a set of configuration attributes (or parameters) that govern IBM MQ.

Change attribute information by editing the IBM MQ configuration files. On IBM MQ for Windows and Linux (x86 and x86-64 platforms), the IBM MQ configuration files can be edited using the MQ Explorer.

On Windows systems you can also use amqmdain to change configuration information, as described in amqmdain

To find out more about configuring IBM MQ and queue managers for your platform, see the following subtopics:

Related concepts

"Configuring" on page 5
Create one or more queue managers on one or more computers, and configure them on your development, test, and production systems to process messages that contain your business data.

"Configuring queue managers on z/OS" on page 522
Use these instructions to configure queue managers on IBM MQ for z/OS.

Related tasks

Planning
Administering IBM MQ

Changing configuration information on Windows, UNIX and Linux systems

Configuration attributes are held in configuration files, at the level of the node and of the queue manager.

On Windows, UNIX and Linux platforms, you can change IBM MQ configuration attributes within:

- An IBM MQ configuration file (**mqs.ini**) to effect changes for IBM MQ on the node as a whole. There is one mqs.ini file for each node.

  See "Attributes for changing IBM MQ configuration information" on page 97 for more information on the stanzas included in **mqs.ini**.
- A queue manager configuration file (**qm.ini**) to effect changes for specific queue managers. There is one qm.ini file for each queue manager on the node.

  See "Changing queue manager configuration information" on page 103 for more information on the stanzas included in **qm.ini**.

Client configuration options are held separately, in the client configuration file.

A configuration file (or ***stanza*** file) contains one or more stanzas, which are groups of lines in the .ini file that together have a common function or define part of a system, such as log functions, channel functions, and installable services.

Because the IBM MQ configuration file is used to locate the data associated with queue managers, a nonexistent or incorrect configuration file can cause some or all MQSC commands to fail. Also, applications cannot connect to a queue manager that is not defined in the IBM MQ configuration file.

Any changes you make to a configuration file usually do not take effect until the next time the queue manager is started.

On Windows and Linux (x86 and x86-64 platforms) systems, you can edit configuration information from the MQ Explorer.

On Windows systems you can also use the the `amqmdain` command to edit the configuration files.

For more information about the configuration options on Windows, UNIX and Linux systems, see the following subtopics:

Related concepts

"Configuring" on page 5
Create one or more queue managers on one or more computers, and configure them on your development, test, and production systems to process messages that contain your business data.

"Changing IBM MQ and queue manager configuration information" on page 80
Change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

Related tasks

Planning

Administering IBM MQ

Related reference

"Changing configuration information on IBM i" on page 86
Use this information to learn how to change the behavior of queue managers to suit your installation's needs.

"Attributes for changing IBM MQ configuration information" on page 97
On IBM MQ for Windows systems and on IBM MQ for Linux (x86 and x86-64 platforms) systems, modify configuration information using the IBM MQ Explorer. On other systems, modify the information by editing the mqs.ini configuration file.

"Changing queue manager configuration information" on page 103
The attributes described here modify the configuration of an individual queue manager. They override any settings for IBM MQ.

Editing configuration files

Edit configuration files using commands or a standard text editor.

Before editing a configuration file, back it up so that you have a copy you can revert to if the need arises.

You can edit configuration files either:

- Automatically, using commands that change the configuration of queue managers on the node
- Manually, using a standard text editor

You can edit the default values in the IBM MQ configuration files after installation.

If you set an incorrect value on a configuration file attribute, the value is ignored and an operator message is issued to indicate the problem. (The effect is the same as missing out the attribute entirely.)

When you create a new queue manager:

- Back up the IBM MQ configuration file
- Back up the new queue manager configuration file

Comments can be included in configuration files by adding a ";" or a "#" character before the comment text. If you want to use a ";" or a "#" character without it representing a comment, you can prefix the character with a "\" character and it will be used as part of the configuration data.

When do you need to edit a configuration file?

Edit a configuration file to recover from backup, move a queue manager, change the default queue manager or to assist IBM support.

You might need to edit a configuration file if, for example:

- You lose a configuration file. (Recover from backup if you can.)

- You need to move one or more queue managers to a new directory.
- You need to change your default queue manager; this could happen if you accidentally delete the existing queue manager.
- You are advised to do so by your IBM Support Center.

Configuration file priorities

The value of an attribute is defined in multiple places. Attributes set in commands take precedence over attributes in configuration files.

The attribute values of a configuration file are set according to the following priorities:

- Parameters entered on the command line take precedence over values defined in the configuration files
- Values defined in the qm.ini files take precedence over values defined in the mqs.ini file

The IBM MQ configuration file, mqs.ini

The IBM MQ configuration file, mqs.ini, contains information relevant to all the queue managers on the node. It is created automatically during installation.

On IBM MQ for UNIX and Linux products, the data directory and log directory are always `/var/mqm` and `/var/mqm/log` respectively.

On Windows systems, the location of the data directory `mqs.ini`, and the location of the log directory, are stored in the registry, as their location can vary.

In addition, on Windows systems, the installation configuration information (contained in `mqinst.ini` on IBM MQ for UNIXand Linux systems) is in the registry, as there is no `mqinst.ini` file on Windows.

The mqs.ini file for Windows systems is given by the WorkPath specified in the `HKLM\SOFTWARE\IBM\WebSphere MQ` key. It contains:

- The names of the queue managers
- The name of the default queue manager
- The location of the files associated with each of them

The supplied `LogDefaults` stanza for a new IBM MQ installation does not contain any explicit values for the attributes. The lack of an attribute means that the default for this value is used upon creation of a new queue manager. The default values are shown for the `LogDefaults` stanza in Figure 5 on page 83. A value of zero for the `LogBufferPages` attribute means 512.

If you require a non-default value, you must explicitly specify that value in the `LogDefaults` stanza.

```
#*******************************************************************#
#* Module Name: mqs.ini                                            *#
#* Type        : IBM MQ Machine-wide Configuration File            *#
#* Function    : Define IBM MQ resources for an entire machine     *#
#*******************************************************************#
#* Notes       :                                                   *#
#* 1) This is the installation time default configuration          *#
#*                                                                 *#
#*******************************************************************#
AllQueueManagers:
#*******************************************************************#
#* The path to the qmgrs directory, below which queue manager data *#
#* is stored                                                       *#
#*******************************************************************#
DefaultPrefix=/var/mqm

LogDefaults:
   LogPrimaryFiles=3
   LogSecondaryFiles=2
   LogFilePages=4096
   LogType=CIRCULAR
   LogBufferPages=0
   LogDefaultPath=/var/mqm/log

QueueManager:
   Name=saturn.queue.manager
   Prefix=/var/mqm
   Directory=saturn!queue!manager
   InstallationName=Installation1

QueueManager:
   Name=pluto.queue.manager
   Prefix=/var/mqm
   Directory=pluto!queue!manager
   InstallationName=Installation2

DefaultQueueManager:
   Name=saturn.queue.manager

ApiExitTemplate:
   Name=OurPayrollQueueAuditor
   Sequence=2
   Function=EntryPoint
   Module=/usr/ABC/auditor
   Data=123

ApiExitCommon:
   Name=MQPoliceman
   Sequence=1
   Function=EntryPoint
   Module=/usr/MQPolice/tmqp
   Data=CheckEverything
```

Figure 5. Example of an IBM MQ configuration file for UNIX systems

Queue manager configuration files, qm.ini

A queue manager configuration file, `qm.ini`, contains information relevant to a specific queue manager.

There is one queue manager configuration file for each queue manager. The qm.ini file is automatically created when the queue manager with which it is associated is created.

V 8.0.0.8 From IBM MQ 8.0.0, Fix Pack 8, the **strmqm** command checks the syntax of the CHANNELS and SSL stanzas in the `qm.ini` file before starting the queue manager fully, which makes it much easier to see what is wrong, and correct it quickly if **strmqm** finds that the `qm.ini` file contains any errors. For more information, see strmqm.

Location of the qm.ini files

On UNIX and Linux systems a `qm.ini` file is held in the root of the directory tree occupied by the queue manager. For example, the path and the name for a configuration file for a queue manager called QMNAME is:

```
/var/mqm/qmgrs/QMNAME/qm.ini
```

On Windows systems the location of the `qm.ini` file is given by the WorkPath specified in the `HKLM\SOFTWARE\IBM\WebSphere MQ` key. For example, the path and the name for a configuration file for a queue manager called QMNAME is:

```
C:\ProgramData\IBM\MQ\qmgrs\QMNAME\qm.ini
```

The queue manager name can be up to 48 characters in length. However, this does not guarantee that the name is valid or unique. Therefore, a directory name is generated based on the queue manager name. This process is known as *name transformation*. For a description, see Understanding IBM MQ file names.

Example qm.ini file

Linux UNIX

The following example shows how groups of attributes might be arranged in a queue manager configuration file in IBM MQ for UNIX and Linux systems.

```
#* Module Name: qm.ini                                              *#
#* Type       : IBM MQ queue manager configuration file             *#
#  Function   : Define the configuration of a single queue manager *#
#*                                                                  *#
#*******************************************************************#
#* Notes      :                                                     *#
#* 1) This file defines the configuration of the queue manager     *#
#*                                                                  *#
#*******************************************************************#

ExitPath:
   ExitsDefaultPath=/var/mqm/exits
   ExitsDefaultPath64=/var/mqm/exits64

Service:
   Name=AuthorizationService
   EntryPoints=14

ServiceComponent:
   Service=AuthorizationService
   Name=MQSeries.UNIX.auth.service
   Module=amqzfu
   ComponentDataSize=0

Log:
   LogPrimaryFiles=3
   LogSecondaryFiles=2
   LogFilePages=4096
   LogType=CIRCULAR
   LogBufferPages=0 1
   LogPath=/var/mqm/log/saturn!queue!manager/

XAResourceManager:
   Name=DB2 Resource Manager Bank
   SwitchFile=/usr/bin/db2swit
   XAOpenString=MQBankDB
   XACloseString=
   ThreadOfControl=THREAD

Channels: 2
   MaxChannels=200
   MaxActiveChannels=100
   MQIBindType=STANDARD

TCP:
   SndBuffSize=0
```

```
   RcvBuffSize=0
   RcvSndBuffSize=0
   RcvRcvBuffSize=0
   ClntSndBuffSize=0
   ClntRcvBuffSize=0
   SvrSndBuffSize=0
   SvrRcvBuffSize=0

QMErrorLog:
   ErrorLogSize=262144
   ExcludeMessage=7234
   SuppressMessage=9001,9002,9202
   SuppressInterval=30

ApiExitLocal:
   Name=ClientApplicationAPIchecker
   Sequence=3
   Function=EntryPoint
   Module=/usr/Dev/ClientAppChecker
   Data=9.20.176.20
```

Note that the `qm.ini` file for Windows includes an additional *AccessMode* stanza:

```
AccessMode:
SecurityGroup=wmq\wmq
```

Notes:

1. The value of zero for `LogBufferPages` gives a value of 512.
2. For more information on the Channel stanza, see "Initialization and configuration files" on page 170.
3. The maximum number of XAResourceManager stanzas is limited to 255. However, you should use only a small number of stanzas to avoid transaction performance degradation.
4. distributed From IBM MQ 8.0, for new queue managers on distributed platforms, the default TCP send and receive buffer sizes in the TCP stanza of the `qm.ini file` are set to be managed by the operating system. New queue managers are automatically created with a default setting of zero for the send and receive buffers as shown in the following example:

   ```
   TCP:
   SndBuffSize=0
   RcvBuffSize=0
   RcvSndBuffSize=0
   RcvRcvBuffSize=0
   ClntSndBuffSize=0
   ClntRcvBuffSize=0
   SvrSndBuffSize=0
   SvrRcvBuffSize=0
   ```

   Before Version 8.0, without manual tuning, these values defaulted to a fixed size 32Kb buffer.

   This change applies to new queue managers only. The TCP send and receive buffer settings for queue managers that are migrated from earlier releases is retained.

   If the TCP buffer size properties are removed from the `qm.ini` file, then the default buffer is set to 32K. You should exercise caution when using this default as 32K might not be an appropriate buffer for all messaging scenarios.

   If the TCP send and receive buffer properties are set to zero, then the OS default values are used. The method for choosing these defaults will vary by operating system but can typically be found in the "tcp" or get/setsockopt() OS manual pages.

   For information on how to manually set the TCP buffer sizes to use operating system default behavior for migrated queue managers, see TCP, LU62, NETBIOS, and SPX.

See "Changing configuration information on Windows, UNIX and Linux systems" on page 80 for information on when your changes take effect.

Related reference

"TCP, LU62, and NETBIOS" on page 114

Use these queue manager properties pages, or stanzas in the qm.ini file, to specify network protocol configuration parameters. They override the default attributes for channels.

Installation configuration file, mqinst.ini

UNIX and Linux systems

The installation configuration file, `mqinst.ini`, contains information about all the IBM MQ installations on a UNIX or Linux system.

The `mqinst.ini` file is in the `/etc/opt/mqm` directory on UNIX and Linux systems. It contains information about which installation, if any, is the primary installation as well as the following information for each installation:

- The installation name
- The installation description
- The installation identifier
- The installation path

This file must not be edited or referenced directly since its format is not fixed, and could change. Instead, use the following commands to create, delete, query, and modify, the values in the mqinst.ini file:

crtmqinst to create entries.
dltmqinst to delete entries.
dspmqinst to display entries.
setmqinst to set entries.

The installation identifier, for internal use only, is set automatically and must not be changed.

Windows systems

Installation configuration information is held in the following key on Windows systems:

```
HKLM\SOFTWARE\IBM\WebSphere MQ\Installation\<InstallationName>
```

This key must not be edited or referenced directly since its format is not fixed, and could change. Instead, use the following commands to query, and modify, the values in the registry:

dspmqinst to display entries.
setmqinst to set entries.

On Windows, the **`crtmqinst`** and **`dltmqinst`** commands are not available. The installation and uninstallation processes handle the creation and deletion of the required registry entries.

Changing configuration information on IBM i

Use this information to learn how to change the behavior of queue managers to suit your installation's needs.

You change IBM MQ configuration information by modifying the values specified on a set of configuration attributes (or parameters) that govern IBM MQ. You change these attributes by editing the **IBM MQ configuration files**.

For information on modifying the configuration values on IBM i, see the following topics:

- "IBM MQ for IBM i configuration files" on page 87
- "Attributes for changing IBM MQ for IBM i configuration information" on page 88
- "Changing IBM MQ queue manager configuration information" on page 91

- “Example IBM i mqs.ini and qm.ini files” on page 95

Related concepts

“Configuring” on page 5
Create one or more queue managers on one or more computers, and configure them on your development, test, and production systems to process messages that contain your business data.

“Changing IBM MQ and queue manager configuration information” on page 80
Change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

“Changing configuration information on Windows, UNIX and Linux systems” on page 80
Configuration attributes are held in configuration files, at the level of the node and of the queue manager.

Related tasks

Planning

Administering IBM MQ

Related reference

“Attributes for changing IBM MQ configuration information” on page 97
On IBM MQ for Windows systems and on IBM MQ for Linux (x86 and x86-64 platforms) systems, modify configuration information using the IBM MQ Explorer. On other systems, modify the information by editing the mqs.ini configuration file.

“Changing queue manager configuration information” on page 103
The attributes described here modify the configuration of an individual queue manager. They override any settings for IBM MQ.

IBM MQ for IBM i configuration files

Use this information to understand the methods for configuring IBM MQ for IBM i.

You modify IBM MQ configuration attributes within:

- An IBM MQ configuration file, `mqs.ini`, effect changes on the node as a whole. There is one `mqs.ini` file for each IBM MQ installation.
- A queue manager configuration file, `qm.ini`, effect changes for specific queue managers. There is one `qm.ini` file for each queue manager on the node.

Note that .ini files are stream files resident in the IFS.

A configuration file (which can be referred to as a ***stanza*** file) contains one or more stanzas, which are groups of lines in the .ini file that together have a common function or define part of a system, for example, log functions and channel functions. Any changes made to a configuration file do not take effect until the next time the queue manager is started.

Editing configuration files

Before editing a configuration file, back it up so that you have a copy you can revert to if the need arises.

You can edit configuration files either:

- Automatically, using commands that change the configuration of queue managers on the node.
- Manually, using the EDTF CL editor.

You can edit the default values in the IBM MQ configuration files after installation. If you set an incorrect value on a configuration file attribute, the value is ignored and an operator message is issued to indicate the problem. (The effect is the same as missing out the attribute entirely.)

When you create a new queue manager:

- Back up the IBM MQ configuration file.
- Back up the new queue manager configuration file.

When do you need to edit a configuration file?

You might need to edit a configuration file if, for example:

- You lose a configuration file; recover from backup if possible.
- You need to move one or more queue managers to a new directory.
- You need to change your default queue manager; this could happen if you accidentally delete the existing queue manager.
- You are advised to do so by your IBM Support Center.

Configuration file priorities

The attribute values of a configuration file are set according to the following priorities:

- Parameters entered on the command line take precedence over values defined in the configuration files.
- Values defined in the `qm.ini` files take precedence over values defined in the `mqs.ini` file.

The IBM MQ configuration file mqs.ini

The IBM MQ configuration file, `mqs.ini`, contains information relevant to all the queue managers on an IBM MQ installation. It is created automatically during installation. In particular, the `mqs.ini` file is used to locate the data associated with each queue manager.

The `mqs.ini` file is stored in `/QIBM/UserData/mqm`

The `mqs.ini` file contains:

- The names of the queue managers.
- The name of the default queue manager.
- The location of the files associated with each queue manager.
- Information identifying any API exits (see Configuring API exits for more information).

Queue manager configuration files qm.ini

A queue manager configuration file, `qm.ini`, contains information relevant to a specific queue manager. There is one queue manager configuration file for each queue manager. The `qm.ini` file is automatically created when the queue manager with which it is associated is created.

A `qm.ini` file is held in the *<mqmdata directory>*/*QMNAME*/`qm.ini`, where *<mqmdata directory>* is `/QIBM/UserData/mqm` by default and *QMNAME* is the name of the queue manager to which the initialization file applies.

Note:

1. You can change the *<mqmdata directory>* in the `mqs.ini` file.
2. The queue manager name can be up to 48 characters in length. However, this does not guarantee that the name is valid or unique. Therefore, a directory name is generated based on the queue manager name. This process is known as **name transformation**. See Understanding IBM MQ for IBM i queue manager library names for further information.

Attributes for changing IBM MQ for IBM i configuration information

Use this information to understand the configuration information stanzas.

The following groups of attributes occur in `mqs.ini`:

- “The AllQueueManagers stanza” on page 89
- “The DefaultQueueManager stanza” on page 90
- “The ExitProperties stanza” on page 90
- “The QueueManager stanza” on page 90

There are also two stanzas associated with API exits, `ApiExitCommon` and `ApiExitTemplate`. For details on using these, see Configuring API exits.

The AllQueueManagers stanza

The `AllQueueManagers` stanza can specify:

- The path to the `qmgrs` directory where the files associated with a queue manager are stored
- The path to the executable library
- The method for converting EBCDIC-format data to ASCII format

In the descriptions of the stanzas, the value underlined is the default value and the | symbol means *or*.

DefaultPrefix= ***directory_name***

The path to the `qmgrs` directory, within which the queue manager data is kept. If you change the default prefix for the queue manager, you must replicate the directory structure that was created at installation time. In particular, you must create the `qmgrs` structure. Stop IBM MQ before changing the default prefix, and restart IBM MQ only after moving the structures to the new location and changing the default prefix.

As an alternative to changing the default prefix, you can use the environment variable MQSPREFIX to override the `DefaultPrefix` for the `CRTMQM` command.

ConvEBCDICNewline=NL_TO_LF|TABLE|ISO

EBCDIC code pages contain a newline (NL) character that is not supported by ASCII code pages, although some ISO variants of ASCII contain an equivalent.

Use the ConvEBCDICNewline attribute to specify the method IBM MQ is to use when converting the EBCDIC NL character into ASCII format.

NL_TO_LF

Convert the EBCDIC NL character (X'15') to the ASCII line feed character, LF (X'0A'), for all EBCDIC to ASCII conversions.

NL_TO_LF is the default.

TABLE

Convert the EBCDIC NL character according to the conversion tables used on IBM i for all EBCDIC to ASCII conversions.

Note that the effect of this type of conversion can vary from language to language .

ISO

Specify ISO if you want:

- ISO CCSIDs to be converted using the TABLE method
- All other CCSIDs to be converted using the NL_TO_CF method.

Possible ISO CCSIDs are shown in Table 11 on page 89.

Table 11. List of possible ISO CCSIDs

CCSID	Code Set
819	ISO8859-1
912	ISO8859-2
915	ISO8859-5
1089	ISO8859-6
813	ISO8859-7
916	ISO8859-8

Table 11. List of possible ISO CCSIDs (continued)	
CCSID	**Code Set**
920	ISO8859-9
1051	roman8

If the ASCII CCSID is not an ISO subset, ConvEBCDICNewline defaults to NL_TO_LF.

The DefaultQueueManager stanza

The `DefaultQueueManager` stanza specifies the default queue manager for the node.

Name= *default_queue_manager*
The default queue manager processes any commands for which a queue manager name is not explicitly specified. The `DefaultQueueManager` attribute is automatically updated if you create a new default queue manager. If you inadvertently create a new default queue manager and then want to revert to the original, you must alter the `DefaultQueueManager` attribute manually.

The ExitProperties stanza

The `ExitProperties` stanza specifies configuration options used by queue manager exit programs.

In the descriptions of the stanzas, the value underlined is the default value and the | symbol means *or*.

CLWLMode= SAFE |FAST
The cluster workload exit, CLWL, allows you to specify which cluster queue in the cluster is to be opened in response to an MQI call (for example: MQOPEN or MQPUT). The CLWL exit runs either in FAST mode or SAFE mode depending on the value you specify on the CLWLMode attribute. If you omit the CLWLMode attribute, the cluster workload exit runs in SAFE mode.

SAFE
Run the CLWL exit in a separate process to the queue manager. This is the default.

If a problem arises with the user-written CLWL exit when running in SAFE mode, the following happens:

- The CLWL server process (amqzlwa0) fails
- The queue manager restarts the CLWL server process
- The error is reported to you in the error log. If an MQI call is in progress, you receive notification in the form of a bad return code.

The integrity of the queue manager is preserved.

Note: Running the CLWL exit in a separate process might have a detrimental effect on performance.

FAST
Run the cluster exit inline in the queue manager process.

Specifying this option improves performance by avoiding the overheads associated with running in SAFE mode, but does so at the expense of queue manager integrity. Run the CLWL exit in FAST mode only if you are convinced that there are **no** problems with your CLWL exit, and you are particularly concerned about performance overheads.

If a problem arises when the CLWL exit is running in FAST mode, the queue manager fails and you run the risk of compromising the integrity of the queue manager.

The QueueManager stanza

There is one `QueueManager` stanza for every queue manager. These attributes specify the queue manager name and the name of the directory containing the files associated with that queue manager.

The name of the directory is based on the queue manager name, but is transformed if the queue manager name is not a valid file name.

See Understanding IBM MQ for IBM i queue manager library names for more information about name transformation.

Name= *queue_manager_name*
The name of the queue manager.

Prefix= *prefix*
Where the queue manager files are stored. By default, this is the same as the value specified on the DefaultPrefix attribute of the `AllQueueManager` stanza in the `mqs.ini` file.

Directory= *name*
The name of the subdirectory under the `<prefix>\QMGRS` directory where the queue manager files are stored. This name is based on the queue manager name, but can be transformed if there is a duplicate name, or if the queue manager name is not a valid file name.

Library= *name*
The name of the library where IBM i objects pertinent to this queue manager, for example, journals and journal receivers, are stored. This name is based on the queue manager name, but can be transformed if there is a duplicate name, or if the queue manager name is not a valid library name.

Changing IBM MQ queue manager configuration information

Use this information to understand the queue manager configuration stanzas.

There are two stanzas associated with API exits, `ApiExitCommon` and `ApiExitTemplate`. For details on using these, see Configuring API exits.

The following groups of attributes can occur in a `qm.ini` file for a specific queue manager, or used to override values set in `mqs.ini`.

See the following topics for changing configuration information for specific options:

- "The Log stanza" on page 91
- "The Channels stanza" on page 92
- "QMErrorLog stanza on IBM i" on page 93
- "The TCP stanza" on page 94
- "PreConnect stanza of the client configuration file" on page 65

The Log stanza

Parameters for configuring the log file.

The `Log` stanza specifies the log attributes for a particular queue manager. By default, these are inherited from the settings specified in the `LogDefaults` stanza in the mqs.ini file when the queue manager is created.

Only change attributes of this stanza if you want to configure a queue manager differently from others.

The values specified on the attributes in the qm.ini file are read when the queue manager is started. The file is created when the queue manager is created.

LogBufferSize
The journal buffer size, in bytes. Enter a number in the range 32 000 through 15 761 440. The default is 32 000.

LogPath= *library_name*
The name of the library used to store journals and journal receivers for this queue manager.

LogReceiverSize
The journal receiver size, in kilobytes. The default is 100 000.

The Channels stanza

The `Channels` stanza contains information about the channels.

MaxChannels= 100 | *number*
The maximum number of *current* channels allowed. For z/OS, the value must be 1 - 9999, with a default value of 200. For all other platforms, the value must be 1 - 65,535, with a default value of 100.

MaxActiveChannels= *MaxChannels_value*
The maximum number of channels allowed to be *active* at any time. The default is the value specified on the `MaxChannels` attribute.

MaxInitiators= 3 | *number*
The maximum number of initiators. The default and maximum value is 3.

MQIBINDTYPE=FASTPATH| STANDARD
The binding for applications.

FASTPATH
Channels connect using MQCONNX FASTPATH. That is, there is no agent process.

STANDARD
Channels connect using STANDARD.

ThreadedListener= NO |YES
Whether to start RUNMQLSR (YES) or AMQCLMAA (NO) as a listener.

If you specify ThreadedListener=YES, all channels run as threads of a single job. This limits the number of connections to the resources available to a single job.

If you specify ThreadedListener=NO, the non-threaded listener (AMQCLMAA) starts a new responder job (AMQCRSTA) for each inbound TCP/IP channel. The disadvantage of this technique is that it is not as fast to start a new AMQCRSTA job as it is to start a thread within a RUNMQLSR job, therefore connection times for a non-threaded listener are slower than those for a threaded listener.

AdoptNewMCA= NO |SVR|SNDR|RCVR|CLUSRCVR|ALL|FASTPATH
If IBM MQ receives a request to start a channel, but finds that an amqcrsta process exists for the same channel, the existing process must be stopped before the new one can start. The `AdoptNewMCA` attribute allows you to control the ending of an existing process and the startup of a new one for a specified channel type.

If you specify the `AdoptNewMCA` attribute for a given channel type, but the new channel fails to start because the channel is already running:

1. The new channel tries to end the previous one.
2. If the previous channel server does not end by the time the AdoptNewMCATimeout wait interval expires, the process (or the thread) for the previous channel server is ended.
3. If the previous channel server has not ended after step 2, and after the AdoptNewMCATimeout wait interval expires for a second time, IBM MQ ends the channel with a `CHANNEL IN USE` error.

You specify one or more values, separated by commas or blanks, from the following list:

NO
The `AdoptNewMCA` feature is not required. This is the default.

SVR
Adopt server channels

SNDR
Adopt sender channels

RCVR
Adopt receiver channels

CLUSRCVR
Adopt cluster receiver channels

ALL
Adopt all channel types, except for FASTPATH channels

FASTPATH
Adopt the channel if it is a FASTPATH channel. This happens only if the appropriate channel type is also specified, for example, AdoptNewMCA=RCVR,SVR,FASTPATH

Attention!: The AdoptNewMCA attribute can behave in an unpredictable fashion with FASTPATH channels because of the internal design of the queue manager. Exercise great caution when enabling the `AdoptNewMCA` attribute for FASTPATH channels.

AdoptNewMCATimeout= 60 |1-3600
The amount of time, in seconds, that the new process waits for the old process to end. Specify a value, in seconds, in the range 1 - 3600. The default value is 60.

AdoptNewMCACheck=QM|ADDRESS|NAME|ALL
The `AdoptNewMCACheck` attribute allows you to specify the type checking required when enabling the `AdoptNewMCA` attribute. It is important for you to perform all three of the following checks, if possible, to protect your channels from being shut down, inadvertently or maliciously. At the very least check that the channel names match.

Specify one or more values, separated by commas or blanks, from the following:

QM
The listener process checks that the queue manager names match.

ADDRESS
The listener process checks the communications address, for example, the TCP/IP address.

NAME
The listener process checks that the channel names match.

ALL
The listener process checks for matching queue manager names, the communications address, and for matching channel names.

The default is `AdoptNewMCACheck=NAME,ADDRESS,QM`.

Related concepts
"Channel states" on page 154
A channel can be in one of many states at any time. Some states also have substates. From a given state a channel can move into other states.

IBM i *QMErrorLog stanza on IBM i*

Use the `QMErrorLog` stanza in the `qm.ini` file to tailor the operation and contents of queue manager error logs.

ErrorLogSize= *maxsize*
Specifies the size of the queue manager error log at which it is copied to the backup. *maxsize* must be in the range 1048576 through 2147483648 bytes. If `ErrorLogSize` is not specified, the default value of 2097152 bytes (2 MB) is used.

ExcludeMessage= *msgIds*
Specifies messages that are not to be written to the queue manager error log. *msqIds* contain a comma separated list of message IDs from the following:

7163 - Job started message (IBM i only)
7234 - Number of messages loaded
9001 - Channel program ended normally
9002 - Channel program started
9202 - Remote host not available
9208 - Error on receive from host
9209 - Connection closed
9228 - Cannot start channel responder
9508 - Cannot connect to queue manager
9524 - Remote queue manager unavailable

9528 - User requested closure of channel
9558 - Remote Channel is not available
9776 - Channel was blocked by user id
9777 - Channel was blocked by NOACCESS map
9782 - Connection was blocked by address
9999 - Channel program ended abnormally

SuppressMessage= *msgIds*
Specifies messages that will be written to the queue manager error log once only in a specified time interval. The time interval is specified by `SuppressInterval` . *msqIds* contain a comma separated list of message IDs from the following:

7163 - Job started message (IBM i only)
7234 - Number of messages loaded
9001 - Channel program ended normally
9002 - Channel program started
9202 - Remote host not available
9208 - Error on receive from host
9209 - Connection closed
9228 - Cannot start channel responder
9508 - Cannot connect to queue manager
9524 - Remote queue manager unavailable
9528 - User requested closure of channel
9558 - Remote Channel is not available
9776 - Channel was blocked by user id
9777 - Channel was blocked by NOACCESS map
9782 - Connection was blocked by address
9999 - Channel program ended abnormally

If the same message id is specified in both `SuppressMessage` and `ExcludeMessage` , the message is excluded.

SuppressInterval= *length*
Specifies the time interval, in seconds, in which messages specified in `SuppressMessage` will be written to the queue manager error log once only. *length* must be in the range 1 through 86400 seconds. If `SuppressInterval` is not specified, the default value of 30 seconds is used.

The TCP stanza

Use these queue manager properties pages, or stanzas in the `qm.ini` file, to specify network protocol configuration parameters. They override the default attributes for channels.

Note: Only attributes representing changes to the default values need to be specified.

TCP
The following attributes can be specified:

Port= <u>1414</u> | *port_number*
The default port number, in decimal notation, for TCP/IP sessions. The default port number for IBM MQ 8.0 is 1414.

KeepAlive= <u>NO</u> |YES
Switch the KeepAlive function on or off. KeepAlive=YES causes TCP/IP to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

ListenerBacklog=number
When receiving on TCP/IP, a maximum number of outstanding connection requests is set. This can be considered to be a *backlog* of requests waiting on the TCP/IP port for the listener to accept the request. The default listener backlog value for IBM i is 255; the maximum is 512. If the backlog reaches the value of 512, the TCP/IP connection is rejected and the channel cannot start.

For MCA channels, this results in the channel going into a RETRY state and retrying the connection at a later time.

For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and should retry the connection at a later time.

The `ListenerBacklog` attribute allows you to override the default number of outstanding requests for the TCP/IP listener.

Connect_Timeout=number| 0
The number of seconds before an attempt to connect the socket times out. The default value of zero specifies that there is no connect timeout.

The following group of properties can be used to control the size of buffers used by TCP/IP. The values are passed directly to the TCP/IP layer of the operating system. Great care should be taken when using these properties. If the values are set incorrectly it can adversely affect the TCP/IP performance. For further information about how this affects performance refer to the TCP/IP documentation for your environment. A value of zero indicates that the operating system will manage the buffer sizes, as opposed to the buffer sizes being fixed by IBM MQ.

SndBuffSize=number| 0
The size in bytes of the TCP/IP send buffer used by the sending end of channels. This stanza value can be overridden by a stanza more specific to the channel type, for example RcvSndBuffSize.

RcvBuffSize=number| 0
The size in bytes of the TCP/IP receive buffer used by the receiving end of channels. This stanza value can be overridden by a stanza more specific to the channel type, for example RcvRcvBuffSize. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

RcvSndBuffSize=number| 0
The size in bytes of the TCP/IP send buffer used by the sender end of a receiver channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

RcvRcvBuffSize=number| 0
The size in bytes of the TCP/IP receive buffer used by the receiving end of a receiver channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

SvrSndBuffSize=number| 0
The size in bytes of the TCP/IP send buffer used by the server end of a client-connection server-connection channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

SvrRcvBuffSize=number| 0
The size in bytes of the TCP/IP receive buffer used by the server end of a client-connection server-connection channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

Example IBM i mqs.ini and qm.ini files

Example IBM MQ configuration file

The following example shows an `mqs.ini` file for IBM i:

```
#*******************************************************************#
#* Module Name: mqs.ini                                            *#
#* Type       : IBM MQ Configuration File                          *#
#* Function   : Define IBM MQ resources for the node               *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This is an example IBM MQ configuration file                 *#
#*                                                                 *#
```

```
#*******************************************************************#
AllQueueManagers:
#*******************************************************************#
#* The path to the qmgrs directory, within which queue manager data    *#
#* is stored                                                           *#
#*******************************************************************#
DefaultPrefix=/QIBM/UserData/mqm

QueueManager:
Name=saturn.queue.manager
Prefix=/QIBM/UserData/mqm
Library=QMSATURN.Q
Directory=saturn!queue!manager

QueueManager:
Name=pluto.queue.manager
Prefix=/QIBM/UserData/mqm
Library=QMPLUTO.QU
Directory=pluto!queue!manager

DefaultQueueManager:
Name=saturn.queue.manager
```

Example queue manager configuration file

The following example shows how groups of attributes might be arranged in a queue manager configuration file for IBM i.

```
#*******************************************************************#
#* Module Name: qm.ini                                             *#
#* Type       : IBM MQ queue manager configuration file            *#
# Function    : Define the configuration of a single queue manager *#
#*                                                                 *#
#*******************************************************************#
#* Notes      :                                                    *#
#* 1) This file defines the configuration of the queue manager     *#
#*                                                                 *#
#*******************************************************************#
Log:
LogPath=QMSATURN.Q
LogReceiverSize=65536

CHANNELS:
MaxChannels = 20       ; Maximum number of channels allowed.
                       ; Default is 100.
MaxActiveChannels = 10 ; Maximum number of channels allowed to be
                       ; active at any time. The default is the
                       ; value of MaxChannels.

TCP:                   ; TCP/IP entries.
KeepAlive = Yes        ; Switch KeepAlive on.
SvrSndBuffSize=20000   ; Size in bytes of the TCP/IP send buffer for each
                       ; channel instance. Default is 32768.
SvrRcvBuffSize=20000   ; Size in bytes of the TCP/IP receive buffer for each
                       ; channel instance. Default is 32768.
Connect_Timeout=10000  ; Number of seconds before an attempt to connect the
                       ; channel instance times out. Default is zero (no timeout).

QMErrorLog:
ErrorLogSize = 262144
ExcludeMessage = 7234
SuppressMessage = 9001,9002,9202
SuppressInterval = 30
```

Notes:

1. IBM MQ on the node uses the default locations for queue managers and the journals.
2. The queue manager saturn.queue.manager is the default queue manager for the node. The directory for files associated with this queue manager has been automatically transformed into a valid file name for the file system.
3. Because the IBM MQ configuration file is used to locate the data associated with queue managers, a nonexistent or incorrect configuration file can cause some or all IBM MQ commands to fail. Also, applications cannot connect to a queue manager that is not defined in the IBM MQ configuration file.

Attributes for changing IBM MQ configuration information

On IBM MQ for Windows systems and on IBM MQ for Linux (x86 and x86-64 platforms) systems, modify configuration information using the IBM MQ Explorer. On other systems, modify the information by editing the mqs.ini configuration file.

See the following subtopics for attributes for specific components:

Related concepts

“Configuring” on page 5
Create one or more queue managers on one or more computers, and configure them on your development, test, and production systems to process messages that contain your business data.

“Changing IBM MQ and queue manager configuration information” on page 80
Change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

Related tasks

Planning

Administering IBM MQ

Related reference

“Changing queue manager configuration information” on page 103
The attributes described here modify the configuration of an individual queue manager. They override any settings for IBM MQ.

All queue managers

Use the `General` and `Extended` IBM MQ properties page from the MQ Explorer, or the `AllQueueManagers` stanza in the mqs.ini file to specify the following information about all queue managers.

DefaultPrefix= ***directory_name***

This attribute specifies the path to the qmgrs directory, within which the queue manager data is kept.

If you change the default prefix for the queue manager, replicate the directory structure that was created at installation time.

In particular, you must create the qmgrs structure. Stop IBM MQ before changing the default prefix, and restart IBM MQ only after you have moved the structures to the new location and changed the default prefix.

Note: Do not delete the `/var/mqm/errors` directory on UNIX and Linux systems, or the `\errors` directory on Windows systems.

As an alternative to changing the default prefix, you can use the environment variable MQSPREFIX to override the `DefaultPrefix` for the `crtmqm` command.

Because of operating system restrictions, keep the supplied path sufficiently short so that the sum of the path length and any queue manager name is a maximum of 70 characters long.

ConvEBCDICNewline=NL_TO_LF|TABLE|ISO

EBCDIC code pages contain a newline (NL) character that is not supported by ASCII code pages (although some ISO variants of ASCII contain an equivalent). Use the ConvEBCDICNewline attribute to specify how IBM MQ is to convert the EBCDIC NL character into ASCII format.

z/OS The ConvEBCDICNewline attribute is not available on z/OS. The behavior on z/OS is equivalent to `ConvEBCDICNewline=TABLE`. Note that the default on other platforms might be different.

NL_TO_LF

Convert the EBCDIC NL character (X'15') to the ASCII line feed character, LF (X'0A'), for all EBCDIC to ASCII conversions.

NL_TO_LF is the default.

TABLE
Convert the EBCDIC NL character according to the conversion tables used on your platform for all EBCDIC to ASCII conversions.

The effect of this type of conversion might vary from platform to platform and from language to language; even on the same platform, the behavior might vary if you use different CCSIDs.

ISO
Convert:

- ISO CCSIDs using the TABLE method
- All other CCSIDs using the NL_TO_CF method

Possible ISO CCSIDs are shown in Table 12 on page 98.

Table 12. List of possible ISO CCSIDs

CCSID	**Code Set**
819	ISO8859-1
912	ISO8859-2
915	ISO8859-5
1089	ISO8859-6
813	ISO8859-7
916	ISO8859-8
920	ISO8859-9
1051	roman8

If the ASCII CCSID is not an ISO subset, ConvEBCDICNewline defaults to NL_TO_LF.

V 8.0.0.12 From IBM MQ 8.0.0, Fix Pack 12, you can use the **`AMQ_CONVEBCDICNEWLINE`** environment variable instead of the **`ConvEBCDICNewline`** stanza attribute, for example to provide **`ConvEBCDICNewline`** functionality on the client side in situations where the `mqs.ini` file cannot be used. The environment variable takes the same values (`NL_TO_LF`, `TABLE`, or `ISO`) as the **`ConvEBCDICNewline`** attribute. The stanza attribute takes precedence if both the attribute and the environment variable are set.

Default queue manager

Use the `General` IBM MQ properties page from the IBM MQ Explorer, or the `DefaultQueueManager` stanza in the mqs.ini file to specify the default queue manager.

Name= *default_queue_manager*
The default queue manager processes any commands for which a queue manager name is not explicitly specified. The `DefaultQueueManager` attribute is automatically updated if you create a new default queue manager. If you inadvertently create a new default queue manager and then want to revert to the original, alter the `DefaultQueueManager` attribute manually.

Exit properties

Use the `Extended` IBM MQ properties page from the IBM MQ Explorer, or the `ExitProperties` stanza in the mqs.ini file to specify configuration options used by queue manager exit programs.

CLWLMode= SAFE |FAST
The cluster workload (CLWL) exit allows you to specify which cluster queue in the cluster to open in response to an MQI call (for example, `MQOPEN` , `MQPUT`). The CLWL exit runs either in FAST mode or SAFE mode depending on the value you specify on the CLWLMode attribute. If you omit the CLWLMode attribute, the cluster workload exit runs in SAFE mode.

SAFE

Run the CLWL exit in a separate process from the queue manager. This is the default.

If a problem arises with the user-written CLWL exit when running in SAFE mode, the following happens:

- The CLWL server process (amqzlwa0) fails.
- The queue manager restarts the CLWL server process.
- The error is reported to you in the error log. If an MQI call is in progress, you receive notification in the form of a return code.

The integrity of the queue manager is preserved.

Note: Running the CLWL exit in a separate process can affect performance.

FAST

Run the cluster exit inline in the queue manager process.

Specifying this option improves performance by avoiding the process switching costs associated with running in SAFE mode, but does so at the expense of queue manager integrity. You should only run the CLWL exit in FAST mode if you are convinced that there are **no** problems with your CLWL exit, and you are particularly concerned about performance.

If a problem arises when the CLWL exit is running in FAST mode, the queue manager will fail and you run the risk of the integrity of the queue manager being compromised.

Log defaults for IBM MQ

Use the `Default log settings` IBM MQ properties page from the IBM MQ Explorer, or the `LogDefaults` stanza in the mqs.ini file to specify information about log defaults for all queue managers.

If the stanza does not exist then the MQ defaults will be used. The log attributes are used as default values when you create a queue manager, but can be overridden if you specify the log attributes on the `crtmqm` command. See **`crtmqm`** for details of this command.

Once a queue manager has been created, the log attributes for that queue manager are taken from the settings described in “Queue manager logs” on page 107.

The default prefix (specified in “All queue managers” on page 97) and log path specified for the particular queue manager (specified in “Queue manager logs” on page 107) allow the queue manager and its log to be on different physical drives. This is the recommended method, although by default they are on the same drive.

For information about calculating log sizes, see “Calculating the size of the log” on page 463.

Note: The limits given in the following parameter list are limits set by IBM MQ. Operating system limits might reduce the maximum possible log size.

LogPrimaryFiles= 3 | *2-254* (Windows)| *2-510* (UNIX and Linux systems)

The log files allocated when the queue manager is created.

The minimum number of primary log files you can have is 2 and the maximum is 254 on Windows, or 510 on UNIX and Linux systems. The default is 3.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX and Linux systems, and must not be less than 3.

The value is examined when the queue manager is created or started. You can change it after the queue manager has been created. However, a change in the value is not effective until the queue manager is restarted, and the effect might not be immediate.

LogSecondaryFiles= 2 | *1-253* (Windows)| *1-509* (UNIX and Linux systems)

The log files allocated when the primary files are exhausted.

The minimum number of secondary log files is 1 and the maximum is 253 on Windows, or 509 on UNIX and Linux systems. The default number is 2.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX and Linux systems, and must not be less than 3.

The value is examined when the queue manager is started. You can change this value, but changes do not become effective until the queue manager is restarted, and even then the effect might not be immediate.

LogFilePages= ***number***

The log data is held in a series of files called log files. The log file size is specified in units of 4 KB pages.

The default number of log file pages is 4096, giving a log file size of 16 MB.

On UNIX and Linux systems the minimum number of log file pages is 64, and on Windows the minimum number of log file pages is 32; in both cases the maximum number is 65 535.

Note: The size of the log files specified during queue manager creation cannot be changed for a queue manager.

LogType= **`CIRCULAR`** **|LINEAR**

The type of log to be used. The default is CIRCULAR.

`CIRCULAR`

Start restart recovery using the log to roll back transactions that were in progress when the system stopped.

See “Types of logging” on page 458 for a fuller explanation of circular logging.

LINEAR

For both restart recovery and media or forward recovery (creating lost or damaged data by replaying the contents of the log).

See “Types of logging” on page 458 for a fuller explanation of linear logging.

If you want to change the default, you can either edit the `LogType` attribute, or specify linear logging using the `crtmqm` command. You cannot change the logging method after a queue manager has been created.

LogBufferPages= **`0`** **|** ***0-4096***

The amount of memory allocated to buffer records for writing, specifying the size of the buffers in units of 4 KB pages.

The minimum number of buffer pages is 18 and the maximum is 4096. Larger buffers lead to higher throughput, especially for larger messages.

If you specify 0 (the default), the queue manager selects the size. In IBM WebSphere MQ 7.1 this is 512 (2048 KB).

If you specify a number in the range 1 through 17, the queue manager defaults to 18 (72 KB). If you specify a number in the range 18 and through 4096, the queue manager uses the number specified to set the memory allocated.

LogDefaultPath= ***directory_name***

The directory in which the log files for a queue manager reside. The directory resides on a local device to which the queue manager can write and, preferably, on a different drive from the message queues. Specifying a different drive gives added protection in case of system failure.

The default is:

- `<DefaultPrefix>\log` for IBM MQ for Windows where `<DefaultPrefix>` is the value specified on the `DefaultPrefix` attribute on the `All Queue Managers` IBM MQ properties page. This value is set at install time.
- `/var/mqm/log` for IBM MQ for UNIX and Linux systems

Alternatively, you can specify the name of a directory on the `crtmqm` command using the -ld flag. When a queue manager is created, a directory is also created under the queue manager directory, and this is used to hold the log files. The name of this directory is based on the queue manager name. This ensures that the log file path is unique, and also that it conforms to any limitations on directory name lengths.

If you do not specify -ld on the `crtmqm` command, the value of the `LogDefaultPath` attribute in the mqs.ini file is used.

The queue manager name is appended to the directory name to ensure that multiple queue managers use different log directories.

When the queue manager is created, a `LogPath` value is created in the log attributes in the configuration information, giving the complete directory name for the queue manager's log. This value is used to locate the log when the queue manager is started or deleted.

LogWriteIntegrity=SingleWrite|DoubleWrite| TripleWrite
The method the logger uses to reliably write log records.

TripleWrite
This is the default method.

Note, that you can select **DoubleWrite**, but if you do so, the system interprets this as **TripleWrite**.

SingleWrite
You should use **SingleWrite**, only if the file-system and device hosting the IBM MQ recovery log explicitly guarantees the atomicity of 4KB writes.

That is, when a write of a 4KB page fails for any reason, the only two possible states are either the before image, or the after image. No intermediate state should be possible.

Note: If there is sufficient concurrency in your persistent workload, there is minimal potential benefit in setting anything other than the default value, **TripleWrite**.

Advanced Configuration and Power Interface (ACPI)

Use the ACPI IBM MQ properties page from the IBM MQ Explorer, to specify how IBM MQ is to behave when the system receives a suspend request.

Windows supports the Advanced Configuration and Power Interface (ACPI) standard. This enables Windows users with ACPI enabled hardware to stop and restart channels when the system enters and resumes from suspend mode.

Note that the settings specified in the ACPI IBM MQ properties page are applied only when the Alert Monitor is running. The Alert Monitor icon is present on the taskbar if the Alert Monitor is running.

DoDialog= Y | N
Displays the dialog at the time of a suspend request.

DenySuspend=Y | N
Denies the suspend request. This is used if DoDialog=N, or if DoDialog=Y and a dialog cannot be displayed, for example, because your notebook lid is closed.

CheckChannelsRunning=Y | N
Checks whether any channels are running. The outcome can determine the outcome of the other settings.

The following table outlines the effect of each combination of these parameters:

DoDialog	DenySuspend	CheckChannels Running	Action
N	N	N	Accept the suspend request.
N	N	Y	Accept the suspend request.
N	Y	N	Deny the suspend request.
N	Y	Y	If any channels are running deny the suspend request; if not accept the request.

Y	N	N	Display the dialog (see Note ; accept the suspend request). This is the default.
Y	N	Y	If no channels are running accept the suspend request; if they are display the dialog (see Note ; accept the request).
Y	Y	N	Display the dialog (Note ; deny the suspend request).
Y	Y	Y	If no channels are running accept the suspend request; if they are display the dialog (Note ; deny the request).

Note: In cases where the action is to display the dialog, if the dialog cannot be displayed (for example because your notebook lid is closed), the DenySuspend option is used to determine whether the suspend request is accepted or denied.

API exits

Use the MQ Explorer or the `amqmdain` command to change the entries for API exits.

Use the `Exits` IBM MQ properties page from the MQ Explorer, or the `ApiExitTemplate` and `ApiExitCommon` stanza in the mqs.ini file to identify API exit routines for all queue managers. On Windows systems, you can also use the `amqmdain` command to change the entries for API exits. (To identify API exit routines for individual queue managers, you use the `ApiExitLocal` stanza, as described in "API exits" on page 117.)

For a complete description of the attributes for these stanzas, see Configuring API exits.

Queue managers

There is one `QueueManager` stanza for every queue manager. Use the stanza to specify the location of the queue manager directory.

On Windows, UNIX and Linux systems, there is one `QueueManager` stanza for every queue manager. These attributes specify the queue manager name, and the name of the directory containing the files associated with that queue manager. The name of the directory is based on the queue manager name, but is transformed if the queue manager name is not a valid file name. See, Understanding IBM MQ file names for more information about name transformation.

Name= ***queue_manager_name***
: The name of the queue manager.

Prefix= ***prefix***
: Where the queue manager files are stored. By default, this value is the same as the value specified on the DefaultPrefix attribute of the All Queue Managers information.

Directory= ***name***
: The name of the subdirectory under the `<prefix>\QMGRS` directory where the queue manager files are stored. This name is based on the queue manager name, but can be transformed if there is a duplicate name or if the queue manager name is not a valid file name.

DataPath= ***path***
: An explicit data path provided when the queue manager was created, this overrides Prefix and Directory as the path to the queue manager data.

InstallationName= ***name***
: The name of the IBM MQ installation associated with this queue manager. Commands from this installation must be used when interacting with this queue manager. If no InstallationName value is present, the queue manager is associated with an installation of IBM MQ earlier than Version 7.1.

Related concepts

"Associating a queue manager with an installation" on page 364

When you create a queue manager, it is automatically associated with the installation that issued the **crtmqm** command. On UNIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

Security

Use the `Security` stanza in the `qm.ini` file to specify options for the Object Authority Manager (OAM).

ClusterQueueAccessControl=RQMName|Xmitq
Set this attribute to check the access control of cluster queues or fully qualified queues hosted on cluster queue managers.

RQMName
The profiles checked for access control of remotely hosted queues are named queues or named queue manager profiles.

Xmitq
The profiles checked for access control of remotely hosted queues are resolved to the `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

`Xmitq` is the default value.

GroupModel=GlobalGroups
This attribute determines whether the OAM checks global groups when determining the group membership of a user on Windows.

The default is not to check global groups.

GlobalGroups
The OAM checks global groups.

With `GlobalGroups` set, the authorization commands, **setmqaut**, **dspmqaut**, and **dmpmqaut** accept global groups names; see the **setmqaut** `-g` parameter.

Note: Setting the `ClusterQueueAcessControl=RQMName` and having a custom implementation of the Authorization Service at less than `MQZAS_VERSION_6` results in the queue manager not starting. In this instance, either set `ClusterQueueAcessControl=Xmitq` or upgrade the custom Authorization Service to `MQZAS_VERSION_6` or greater.

Changing queue manager configuration information

The attributes described here modify the configuration of an individual queue manager. They override any settings for IBM MQ.

On UNIX and Linux systems, you modify queue manager configuration information by editing the `qm.ini` configuration file. When you are defining a stanza in `qm.ini`, you do not need to start each item on a new line. You can use either a semicolon (;) or a hash character (#) to indicate a comment.

On Windows and Linux x86-64 systems, you can modify some configuration information by using the MQ Explorer. However, because there are significant implications to changing installable services and their components, the installable services are read-only in the MQ Explorer. You must therefore make any changes to installable services by using **regedit** on Windows, and by editing the `qm.ini` file on UNIX and Linux.

For more details on changing queue manager configuration information, see the following subtopics:

Related concepts

"Configuring" on page 5
Create one or more queue managers on one or more computers, and configure them on your development, test, and production systems to process messages that contain your business data.

"Changing IBM MQ and queue manager configuration information" on page 80
Change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

Related tasks

Planning

Administering IBM MQ

Related reference

"Attributes for changing IBM MQ configuration information" on page 97

On IBM MQ for Windows systems and on IBM MQ for Linux (x86 and x86-64 platforms) systems, modify configuration information using the IBM MQ Explorer. On other systems, modify the information by editing the mqs.ini configuration file.

Access Mode

`Access Mode` applies to Windows servers only. The AccessMode stanza is set by the `-a [r]` option on the **`crtmqm`** command. Do not change the AccessMode stanza after the queue manager has been created.

Use the access group (`-a [r]`) option of the **`crtmqm`** command to specify a Windows security group, members of which will be granted full access to all queue manager data files. The group can either be a local or global group, depending upon the syntax used. Valid syntax for the group name is as follows:

LocalGroup
Domain name\GlobalGroup name
GlobalGroup name @ Domain name

You must define the additional access group before running the crtmqm command with the `-a [r]` option.

If you specify the group using `-ar` instead of `-a` , the local mqm group is not granted access to the queue manager data files. Use this option, if the file system hosting the queue manager data files does not support access control entries for locally defined groups.

The group is typically a global security group, which is used to provide multi-instance queue managers with access to a shared queue manager data and logs folder. Use the additional security access group to set read and write permissions on the folder or to share containing queue manager data and log files.

The additional security access group is an alternative to using the local group named `mqm` to set permissions on the folder containing queue manager data and logs. Unlike the local group `mqm`, you can make the additional security access group a local or a global group. It must be a global group to set permissions on the shared folders that contain the data and log files used by multi-instance queue managers.

The Windows operating system checks the access permissions to read and write queue manager data and log files. It checks the permissions of the user ID that is running queue manager processes. The user ID that is checked depends on whether you started the queue manager as a service or you started it interactively. If you started the queue manager as a service, the user ID checked by the Windows system is the user ID you configured with the **Prepare** IBM MQ wizard. If you started the queue manager interactively, the user ID checked by the Windows system is the user ID that ran the **`strmqm`** command.

The user ID must be a member of the local `mqm` group to start the queue manager. If the user ID is a member of the additional security access group, the queue manager can read and write files that are given permissions by using the group.

Restriction: You can specify an additional security access group only on Windows operating system. If you specify an additional security access group on other operating systems, the **`crtmqm`** command returns an error.

Related concepts

"Secure unshared queue manager data and log directories and files on Windows" on page 434

"Secure shared queue manager data and log directories and files on Windows" on page 431

Related tasks

"Create a multi-instance queue manager on domain workstations or servers" on page 407

Related reference

crtmqm

Installable services

You change installable services on Windows by using **`regedit`**, and on UNIX and Linux by using the `Service` stanza in the `qm.ini` file.

Note: There are significant implications to changing installable services and their components. For this reason, the installable services are read-only in the MQ Explorer.

To change installable services on Windows systems, use **`regedit`**, or on UNIX and Linux systems, use the `Service` stanza in the `qm.ini` file. For each component within a service, you must also specify the name and path of the module containing the code for that component. On UNIX and Linux systems, use the `ServiceComponent` stanza for this.

Name= <u>`AuthorizationService`</u> |NameService

The name of the required service.

<u>`AuthorizationService`</u>

For IBM MQ, the Authorization Service component is known as the object authority manager, or OAM. The `AuthorizationService` stanza and its associated `ServiceComponent` stanza are added automatically when the queue manager is created. Add other `ServiceComponent` stanzas manually.

NameService

No name service is provided by default. If you require a name service, you must add the `NameService` stanza manually.

EntryPoints= *number-of-entries*

The number of entry points defined for the service.

This includes the initialization and termination entry points.

Windows **SecurityPolicy= <u>`Default`</u> |NTSIDsRequired**

On Windows systems, the SecurityPolicy attribute applies only if the service specified is the default authorization service, that is, the OAM. The SecurityPolicy attribute allows you to specify the security policy for each queue manager.

The possible values are:

<u>`Default`</u>

Use the default security policy to take effect. If a Windows security identifier (NT SID) is not passed to the OAM for a particular user ID, an attempt is made to obtain the appropriate SID by searching the relevant security databases.

NTSIDsRequired

Pass an NT SID to the OAM when performing security checks.

See Windows security identifiers (SIDs) for more information.

See also Configuring authorization service stanzas: Windows systems.

Linux UNIX **SecurityPolicy=user|group| <u>`default`</u>**

On UNIX and Linux systems the value specifies whether the queue manager uses user-based or group-based authorization. Values are not case sensitive.

If you do not include this attribute, `default` is used, which uses group-based authorization. Restart the queue manager for changes to become effective. See also Configuring authorization service stanzas: UNIX and Linux systems.

SharedBindingsUserId= *user-type*

The SharedBindingsUserId attribute applies only if the service specified is the default authorization service, that is, the OAM. The SharedBindingsUserId attribute is used with relation to shared bindings only. This value allows you to specify whether the *UserIdentifier* field in the *IdentityContext* structure, from the MQZ_AUTHENTICATE_USER function, is the effective user Id or the real user Id.

For information on the MQZ_AUTHENTICATE_USER function, see MQZ_AUTHENTICATE_USER - Authenticate user.

The possible values are:

Default
: The value of the *UserIdentifier* field is set as the real user Id.

Real
: The value of the *UserIdentifier* field is set as the real user Id.

Effective
: The value of the *UserIdentifier* field is set as the effective user Id.

FastpathBindingsUserId= ***user-type***

The FastpathBindingsUserId attribute applies only if the service specified is the default authorization service, that is, the OAM. The FastpathBindingsUserId attribute is used with relation to fastpath bindings only. This value allows you to specify whether the *UserIdentifier* field in the *IdentityContext* structure, from the MQZ_AUTHENTICATE_USER function, is the effective user Id or the real user Id.

For information on the MQZ_AUTHENTICATE_USER function, see MQZ_AUTHENTICATE_USER - Authenticate user.

The possible values are:

Default
: The value of the *UserIdentifier* field is set as the real user ID.

Real
: The value of the *UserIdentifier* field is set as the real user ID.

Effective
: The value of the *UserIdentifier* field is set as the effective user ID.

IsolatedBindingsUserId= ***user-type***

The **IsolatedBindingsUserId** attribute applies only if the service specified is the default authorization service, that is, the OAM. The **IsolatedBindingsUserId** attribute is used with relation to isolated bindings only. This value allows you to specify whether the *UserIdentifier* field in the *IdentityContext* structure, from the MQZ_AUTHENTICATE_USER function, is the effective user Id or the real user Id.

For information on the MQZ_AUTHENTICATE_USER function, see MQZ_AUTHENTICATE_USER - Authenticate user.

The possible values are:

Default
: The value of the *UserIdentifier* field is set as the effective user Id.

Real
: The value of the *UserIdentifier* field is set as the real user Id.

Effective
: The value of the *UserIdentifier* field is set as the effective user Id.

For more information about installable services and components, see Installable services and components for UNIX, Linux and Windows .

For more information about security services in general, see Setting up security on UNIX and Linux systems.

Related reference

Installable services reference information

Service components

You must specify service component information when you add a new installable service. On Windows systems use `regedit` , and on UNIX and Linux systems use the `ServiceComponent` stanza in the qm.ini file. The authorization service stanza is present by default, and the associated component, the OAM, is active.

Specify the service components as follows:

Service= ***service_name***
The name of the required service. This must match the value specified on the Name attribute of the Service configuration information.

Name= ***component_name***
The descriptive name of the service component. This must be unique and contain only characters that are valid for the names of IBM MQ objects (for example, queue names). This name occurs in operator messages generated by the service. We recommend that this name begins with a company trademark or similar distinguishing string.

Module= ***module_name***
The name of the module to contain the code for this component. This must be a full path name.

ComponentDataSize= ***size***
The size, in bytes, of the component data area passed to the component on each call. Specify zero if no component data is required.

For more information about installable services and components, see Installable services and components for UNIX, Linux and Windows .

Queue manager logs

Use the Log queue manager properties page from the IBM MQ Explorer, or the Log stanza in the qm.ini file, to specify information about logging on a queue manager.

By default, these settings are inherited from the settings specified for the default log settings for the queue manager (described in "Log defaults for IBM MQ" on page 99). Change these settings only if you want to configure this queue manager in a different way.

For information about calculating log sizes, see "Calculating the size of the log" on page 463.

Note: The limits given in the following parameter list are set by IBM MQ. Operating system limits might reduce the maximum possible log size.

LogPrimaryFiles= **3** **|** ***2-254*** **(Windows)|** ***2-510*** **(UNIX and Linux systems)**
The log files allocated when the queue manager is created.

The minimum number of primary log files you can have is 2 and the maximum is 254 on Windows, or 510 on UNIX and Linux systems. The default is 3.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX and Linux systems, and must not be less than 3.

The value is examined when the queue manager is created or started. You can change it after the queue manager has been created. However, a change in the value is not effective until the queue manager is restarted, and the effect might not be immediate.

LogSecondaryFiles= **2** **|** ***1-253*** **(Windows)|** ***1-509*** **(UNIX and Linux systems)**
The log files allocated when the primary files are exhausted.

The minimum number of secondary log files is 1 and the maximum is 253 on Windows, or 509 on UNIX and Linux systems. The default number is 2.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX and Linux systems, and must not be less than 3.

The value is examined when the queue manager is started. You can change this value, but changes do not become effective until the queue manager is restarted, and even then the effect might not be immediate.

LogFilePages= ***number***
The log data is held in a series of files called log files. The log file size is specified in units of 4 KB pages.

The default number of log file pages is 4096, giving a log file size of 16 MB.

On UNIX and Linux systems the minimum number of log file pages is 64, and on Windows the minimum number of log file pages is 32; in both cases the maximum number is 65 535.

Note: The size of the log files specified during queue manager creation cannot be changed for a queue manager.

LogType= CIRCULAR |LINEAR
The type of logging to be used by the queue manager. You cannot change the type of logging to be used once the queue manager has been created. Refer to the description of the LogType attribute in "Log defaults for IBM MQ" on page 99 for information about creating a queue manager with the type of logging you require.

CIRCULAR
Start restart recovery using the log to roll back transactions that were in progress when the system stopped.

See "Types of logging" on page 458 for a fuller explanation of circular logging.

LINEAR
For both restart recovery and media or forward recovery (creating lost or damaged data by replaying the contents of the log).

See "Types of logging" on page 458 for a fuller explanation of linear logging.

LogBufferPages= 0 | *0-4096*
The amount of memory allocated to buffer records for writing, specifying the size of the buffers in units of 4 KB pages.

The minimum number of buffer pages is 18 and the maximum is 4096. Larger buffers lead to higher throughput, especially for larger messages.

If you specify 0 (the default), the queue manager selects the size. In IBM WebSphere MQ 7.1 this is 512 (2048 KB).

If you specify a number in the range 1 through 17, the queue manager defaults to 18 (72 KB). If you specify a number in the range 18 through 4096, the queue manager uses the number specified to set the memory allocated.

The value is examined when the queue manager is started. The value can be increased or decreased within the limits stated. However, a change in the value is not effective until the next time the queue manager is started.

LogPath= *directory_name*
The directory in which the log files for a queue manager reside. This must exist on a local device to which the queue manager can write and, preferably, on a different drive from the message queues. Specifying a different drive gives added protection in case of system failure.

The default is:

- `C:\ProgramData\IBM\MQ\log` in IBM MQ for Windows.
- `/var/mqm/log` in IBM MQ for UNIX and Linux systems.

You can specify the name of a directory on the `crtmqm` command using the -ld flag. When a queue manager is created, a directory is also created under the queue manager directory, and this is used to hold the log files. The name of this directory is based on the queue manager name. This ensures that the log file path is unique, and also that it conforms to any limitations on directory name lengths.

If you do not specify -ld on the `crtmqm` command, the value of the `LogDefaultPath` attribute is used.

In IBM MQ for UNIX and Linux systems, user ID mqm and group mqm must have full authorities to the log files. If you change the locations of these files, you must give these authorities yourself. This is not required if the log files are in the default locations supplied with the product.

LogWriteIntegrity=SingleWrite|DoubleWrite| TripleWrite
The method the logger uses to reliably write log records.

`TripleWrite`
This is the default method.

Note, that you can select **DoubleWrite**, but if you do so, the system interprets this as **TripleWrite**.

SingleWrite
You should use **SingleWrite**, only if the file-system and device hosting the IBM MQ recovery log explicitly guarantees the atomicity of 4KB writes.

That is, when a write of a 4KB page fails for any reason, the only two possible states are either the before image, or the after image. No intermediate state should be possible.

Note: If there is sufficient concurrency in your persistent workload, there is minimal potential benefit in setting anything other than the default value, **TripleWrite**.

Restricted mode

This option applies to UNIX and Linux systems only. The `RestrictedMode` stanza is set by the -g option on the **`crtmqm`** command. Do not change this stanza after the queue manager has been created. If you do not use the -g option, the stanza is not created in the qm.ini file.

There are some directories under which IBM MQ applications create files while they are connected to the queue manager within the queue manager data directory. In order for applications to create files in these directories, they are granted world write access:

- `/var/mqm/sockets/`*`QMgrName`*`/@ipcc/ssem/`*`hostname`*`/`
- `/var/mqm/sockets/`*`QMgrName`*`/@app/ssem/`*`hostname`*`/`
- `/var/mqm/sockets/`*`QMgrName`*`/zsocketapp/`*`hostname`*`/`

where *`<QMGRNAME>`* is the name of the queue manager, and *`<hostname>`* is the host name.

On some systems, it is unacceptable to grant all users write access to these directories. For example, those users who do not need access the queue manager. Restricted mode modifies the permissions of the directories that store queue manager data. The directories can then only be accessed by members of the specified application group. The permissions on the System V IPC shared memory used to communicate with the queue manager are also modified in the same way.

The application group is the name of the group with members that have permission to do the following things:

- run MQI applications
- update all IPCC resources
- change the contents of some queue manager directories

To use restricted mode for a queue manager:

- The creator of the queue manager must be in the `mqm` group and in the application group.
- The `mqm` user ID must be in the application group.
- All users who want to administer the queue manager must be in the `mqm` group and in the application group.
- all users who want to run IBM MQ applications must be in the application group.

Any MQCONN or MQCONNX call issued by a user who is not in the application group failed with reason code MQRC_Q_MGR_NOT_AVAILABLE.

Restricted mode operates with the IBM MQ authorization service. Therefore you must also grant users the authority to connect to IBM MQ and access the resources they require using the IBM MQ authorization service.

Windows Linux UNIX Further information about configuring the IBM MQ authorization service can be found in Setting up security on Windows, UNIX and Linux systems.

Only use IBM MQ restricted mode when the control provided by the authorization service does not provide sufficient isolation of queue manager resources.

XA resource managers

Use the `XA resource manager` queue manager properties page from the IBM MQ Explorer, or the `XAResourceManager` stanza in the qm.ini file, to specify the following information about the resource managers involved in global units of work coordinated by the queue manager.

Add XA resource manager configuration information manually for each instance of a resource manager participating in global units of work; no default values are supplied.

See Database coordination for more information about resource manager attributes.

Name= *name* (mandatory)
This attribute identifies the resource manager instance.

The `Name` value can be up to 31 characters in length. You can use the name of the resource manager as defined in its XA-switch structure. However, if you are using more than one instance of the same resource manager, you must construct a unique name for each instance. You can ensure uniqueness by including the name of the database in the `Name` string, for example.

IBM MQ uses the `Name` value in messages and in output from the `dspmqtrn` command.

Do not change the name of a resource manager instance, or delete its entry from the configuration information, once the associated queue manager has started and the resource manager name is in effect.

SwitchFile= *name* (mandatory)
The fully-qualified name of the load file containing the resource manager's XA switch structure.

If you are using a 64-bit queue manager with 32-bit applications, the `name` value should contain only the base name of the load file containing the resource manager's XA switch structure.

The 32-bit file will be loaded into the application from the path specified by `ExitsDefaultPath` .

The 64-bit file will be loaded into the queue manager from the path specified by `ExitsDefaultPath64` .

XAOpenString= *string* (optional)
The string of data to be passed to the resource manager's xa_open entry point. The contents of the string depend on the resource manager itself. For example, the string could identify the database that this instance of the resource manager is to access. For more information about defining this attribute, see:

- Adding resource manager configuration information for Db2
- Adding resource manager configuration information for Oracle
- Adding resource manager configuration information for Sybase
- Adding resource manager configuration information for Informix®

and consult your resource manager documentation for the appropriate string.

XACloseString= *string* (optional)
The string of data to be passed to the resource manager's xa_close entry point. The contents of the string depend on the resource manager itself. For more information about defining this attribute, see:

- Adding resource manager configuration information for Db2
- Adding resource manager configuration information for Oracle
- Adding resource manager configuration information for Sybase
- Adding resource manager configuration information for Informix

and consult your database documentation for the appropriate string.

ThreadOfControl=THREAD| PROCESS

This attribute is mandatory for IBM MQ for Windows. The queue manager uses this value for serialization when it needs to call the resource manager from one of its own multithreaded processes.

THREAD

The resource manager is fully *thread aware*. In a multithreaded IBM MQ process, XA function calls can be made to the external resource manager from multiple threads at the same time.

PROCESS

The resource manager is not *thread safe*. In a multithreaded IBM MQ process, only one XA function call at a time can be made to the resource manager.

The `ThreadOfControl` entry does not apply to XA function calls issued by the queue manager in a multithreaded application process. In general, an application that has concurrent units of work on different threads requires this mode of operation to be supported by each of the resource managers.

Attributes of the channels stanza

These attributes determine the configuration of a channel.

This information is not applicable to IBM MQ for the z/OS platform.

Use the `Channels` queue manager properties page from the IBM MQ Explorer, or the `CHANNELS` stanza in the `qm.ini` file, to specify information about channels.

MaxChannels= 100 | *number*

The maximum number of *current* channels allowed.

The value must be in the range 1 - 65535. The default is 100.

MaxActiveChannels= *MaxChannels_value*

The maximum number of channels allowed to be *active* at any time. The default is the value specified for the `MaxChannels` attribute.

MaxInitiators= 3 | *number*

The maximum number of initiators. The default and maximum value is 3.

MQIBindType=FASTPATH| STANDARD

The binding for applications:

FASTPATH

Channels connect using `MQCONNX` FASTPATH; there is no agent process.

STANDARD

Channels connect using STANDARD.

PipeLineLength= 1 | *number*

The maximum number of concurrent threads a channel will use. The default is 1. Any value greater than 1 is treated as 2.

When you use pipelining, configure the queue managers at both ends of the channel to have a *PipeLineLength* greater than 1.

Note: Pipelining is only effective for TCP/IP channels.

AdoptNewMCA= NO |SVR|SDR|RCVR|CLUSRCVR|ALL|FASTPATH

If IBM MQ receives a request to start a channel, but finds that an instance of the channel is already running, in some cases the existing channel instance must be stopped before the new one can start. The `AdoptNewMCA` attribute allows you to control which types of channels can be ended in this way.

If you specify the `AdoptNewMCA` attribute for a particular channel type, but the new channel fails to start because a matching channel instance is already running:

1. The new channel tries to stop the previous one by requesting it to end.
2. If the previous channel server does not respond to this request by the time the AdoptNewMCATimeout wait interval expires, the thread or process for the previous channel server is ended.

3. If the previous channel server has not ended after step 2, and after the AdoptNewMCATimeout wait interval expires for a second time, IBM MQ ends the channel with a `CHANNEL IN USE` error.

The AdoptNewMCA functionality applies to server, sender, receiver, and cluster-receiver channels. In the case of a sender or server channel, only one instance of a channel with a particular name can be running in the receiving queue manager. In the case of a receiver or cluster-receiver channel, multiple instances of a channel with a particular name might be running in the receiving queue manager, but only one instance can run at any one time from a particular remote queue manager.

Note: AdoptNewMCA is not supported on requester or server-connection channels.

Specify one or more values, separated by commas or blanks, from the following list:

NO
The `AdoptNewMCA` feature is not required. This is the default.

SVR
Adopt server channels.

SDR
Adopt sender channels.

RCVR
Adopt receiver channels.

CLUSRCVR
Adopt cluster receiver channels.

ALL
Adopt all channel types except FASTPATH channels.

FASTPATH
Adopt the channel if it is a FASTPATH channel. This happens only if the appropriate channel type is also specified, for example: `AdoptNewMCA=RCVR,SVR,FASTPATH`.

Attention!: The AdoptNewMCA attribute might behave in an unpredictable fashion with FASTPATH channels. Exercise great caution when enabling the AdoptNewMCA attribute for FASTPATH channels.

AdoptNewMCATimeout= 60 |1 - 3600
The amount of time, in seconds, that the new channel instance waits for the old channel instance to end. Specify a value in the range 1 - 3600. The default value is 60.

AdoptNewMCACheck=QM|ADDRESS|NAME|ALL
The type of checking required when enabling the `AdoptNewMCA` attribute. If possible, perform full checking to protect your channels from being shut down, inadvertently or maliciously. At the very least, check that the channel names match.

Specify one or more of the following values, separated by commas or blanks in the case of *QM*, *NAME*, or *ALL*:

QM
Check that the queue manager names match.

Note that the queue manager name itself is matched, not the QMID.

ADDRESS
Check the communications source IP address. For example, the TCP/IP address.

Note: Comma separated CONNAME values apply to target addresses and are, therefore, not relevant to this option.

In the case that a multi-instance queue manager fails over from `hosta` to `hostb`, any outbound channels from that queue manager will use the source IP address of `hostb`. If this is different from `hosta`, then AdoptNewMCACheck=*ADDRESS* fails to match.

You can use SSL or TLS with mutual authentication to prevent an attacker from disrupting an existing running channel. Alternatively, use an HACMP type solution with IP-takeover instead of multi-instance queue managers, or use a network load balancer to mask the source IP address.

NAME
: Check that the channel names match.

ALL
: Check for matching queue manager names, the communications address, and for matching channel names.

The default is `AdoptNewMCACheck=NAME,ADDRESS,QM`.

V 8.0.0.5 **ChlauthEarlyAdopt=Y|N|E**

When you use the **`ADOPTCTX(YES)`** parameter on an authentication information object, the security context is set as the user ID that is presented in the MQCSP structure, when validated by a password. In this case, another security context cannot be adopted, unless you set the **`ChlauthEarlyAdopt`** parameter.

Valid values for **`ChlauthEarlyAdopt`** are the following values:

Y
: The channel validates and adopts user ID and password credentials that have been provided by an application using queue manager connection authentication before applying channel authentication rules. In this mode of operation, channel authentication rules match against the user ID resulting from connection authentication checks.

N
: The channel delays connection authentication validation of user ID and password credentials that have been provided by an application until after channel authentication rules have been applied. Note that in this mode of operation, channel authentication blocking and mapping rules cannot consider the results of user ID and password validation.

E
: V 8.0.0.5 From IBM MQ 8.0.0, Fix Pack 5, when security exits are enabled for a channel, allow the adoption of another security context when you use the **`ADOPTCTX(YES)`** parameter in an authentication information object. If you use this value when security exits are not in use, this value is the same as Y.

: V 8.0.0.7 From IBM MQ 8.0.0, Fix Pack 7, the behavior is the same as the value **Y**.

For example, the default authentication information object is set to **`ADOPTCTX(YES)`**, and the user `fred` is logged in. The following two CHLAUTH rules are configured:

```
SET CHLAUTH('MY.CHLAUTH') TYPE(ADDRESSMAP) DESCR('Block all access by
default') ADDRESS('*') USERSRC(NOACCESS) ACTION(REPLACE)
SET CHLAUTH('MY.CHLAUTH') TYPE(USERMAP) DESCR('Allow user bob and force
CONNAUTH') CLNTUSER('bob') CHCKCLNT(REQUIRED) USERSRC(CHANNEL)
```

The following command is issued, with the intention of authenticating the command as the adopted security context of the user bob:

```
runmqsc -c -u bob QMGR
```

In fact, the queue manager uses the security context of `fred`, not bob, and the connection fails.

To use the security context of bob, **`ChlauthEarlyAdopt`** must be set to Y.

`PasswordProtection` = `Compatible` | `always` | `optional`

From IBM MQ 8.0, set protected passwords in the MQCSP structure, rather than using SSL or TLS.

MQCSP password protection is useful for test and development purposes as using MQCSP password protection is simpler than setting up SSL/TLS encryption, but not as secure.

For more information, see MQCSP password protection.

ChlauthIssueWarn = y

Set this attribute if you want message AMQ9787 to be generated when you set the WARN = YES attribute on the SET CHLAUTH command.

Related concepts

"Channel states" on page 154
A channel can be in one of many states at any time. Some states also have substates. From a given state a channel can move into other states.

TCP, LU62, and NETBIOS

Use these queue manager properties pages, or stanzas in the qm.ini file, to specify network protocol configuration parameters. They override the default attributes for channels.

TCP

Use the TCP queue manager properties page from the IBM MQ Explorer, or the TCP stanza in the qm.ini file, to specify Transmission Control Protocol/Internet Protocol (TCP/IP) configuration parameters.

Port= 1414 | *port_number*

The default port number, in decimal notation, for TCP/IP sessions. The *well known* port number for IBM MQ is 1414.

Library1= *DLLName1* (IBM MQ for Windows only)

The name of the TCP/IP sockets DLL.

The default is WSOCK32.

KeepAlive= NO |YES

Switch the KeepAlive function on or off. KeepAlive=YES causes TCP/IP to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

ListenerBacklog=number

Override the default number of outstanding requests for the TCP/IP listener.

When receiving on TCP/IP, a maximum number of outstanding connection requests is set. This can be considered to be a backlog of requests waiting on the TCP/IP port for the listener to accept the request. The default listener backlog values are shown in Table 13 on page 114.

Table 13. Default outstanding connection requests (TCP)

Platform	Default ListenerBacklog value
Windows Server	100
Windows Workstation	5
Linux	100
Solaris	100
HP-UX	20
AIX V5.3 or later	100

Note: Some operating systems support a larger value than the default shown. Use this to avoid reaching the connection limit.

Conversely, some operating systems might limit the size of the TCP backlog, so the effective TCP backlog could be smaller than requested here.

If the backlog reaches the values shown in Table 13 on page 114, the TCP/IP connection is rejected and the channel cannot start. For message channels, this results in the channel going into a RETRY state and retrying the connection at a later time. For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from `MQCONN` and retries the connection at a later time.

The following group of properties can be used to control the size of buffers used by TCP/IP. The values are passed directly to the TCP/IP layer of the operating system. Great care should be taken when using these properties. If the values are set incorrectly it can adversely affect the

TCP/IP performance. For further information about how this affects performance refer to the TCP/IP documentation for your environment. A value of zero indicates that the operating system will manage the buffer sizes, as opposed to the buffer sizes being fixed by IBM MQ.

Connect_Timeout= 0 |number
: The number of seconds before an attempt to connect the socket times out. The default value of zero specifies that there is no connect timeout.

  IBM MQ channel processes connect over nonblocking sockets. Therefore, if the other end of the socket is not ready, connect() returns immediately with *EINPROGRESS* or *EWOULDBLOCK* . Following this, connect will be attempted again, up to a total of 20 such attempts, when a communications error is reported.

  If `Connect_Timeout` is set to a non-zero value, IBM MQ waits for the stipulated period over select() call for the socket to get ready. This increases the chances of success of a subsequent connect() call. This option might be beneficial in situations where connects would require some waiting period, due to high load on the network.

SndBuffSize=number| 0
: The size in bytes of the TCP/IP send buffer used by the sending end of channels. This stanza value can be overridden by a stanza more specific to the channel type, for example RcvSndBuffSize. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

  distributed From Version 8.0, new queue managers are automatically created with a default setting of 0 (see "Queue manager configuration files, qm.ini" on page 83.

RcvBuffSize=number| 0
: The size in bytes of the TCP/IP receive buffer used by the receiving end of channels. This stanza value can be overridden by a stanza more specific to the channel type, for example RcvRcvBuffSize. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

  distributed From Version 8.0, new queue managers are automatically created with a default setting of 0 (see "Queue manager configuration files, qm.ini" on page 83.

RcvSndBuffSize=number| 0
: The size in bytes of the TCP/IP send buffer used by the sender end of a receiver channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

  distributed From Version 8.0, new queue managers are automatically created with a default setting of 0 (see "Queue manager configuration files, qm.ini" on page 83.

RcvRcvBuffSize=number| 0
: The size in bytes of the TCP/IP receive buffer used by the receiving end of a receiver channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

  distributed From Version 8.0, new queue managers are automatically created with a default setting of 0 (see "Queue manager configuration files, qm.ini" on page 83.

SvrSndBuffSize=number| 0
: The size in bytes of the TCP/IP send buffer used by the server end of a client-connection server-connection channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

  distributed From Version 8.0, new queue managers are automatically created with a default setting of 0 (see "Queue manager configuration files, qm.ini" on page 83.

SvrRcvBuffSize=number| 0
: The size in bytes of the TCP/IP receive buffer used by the server end of a client-connection server-connection channel. If the value is set as zero, the operating system defaults are used. If no value is set, then the IBM MQ default, 32768, is used.

distributed From Version 8.0, new queue managers are automatically created with a default setting of 0 (see “Queue manager configuration files, qm.ini” on page 83.

Windows LU62 (IBM MQ for Windows only)

Use the `LU6.2` queue manager properties page from the IBM MQ Explorer, or the `LU62` stanza in the qm.ini file, to specify SNA LU 6.2 protocol configuration parameters.

TPName
The TP name to start on the remote site.

Library1= ***DLLName 1***
The name of the APPC DLL.

The default value is WCPIC32.

Library2= ***DLLName2***
The same as Library1, used if the code is stored in two separate libraries.

The default value is WCPIC32.

Windows NETBIOS (IBM MQ for Windows only)

Use the `Netbios` queue manager properties page from the IBM MQ Explorer, or the `NETBIOS` stanza in the qm.ini file, to specify NetBIOS protocol configuration parameters.

LocalName= ***name***
The name by which this machine is known on the LAN.

AdapterNum= **<u>0</u>** **|** ***adapter_number***
The number of the LAN adapter. The default is adapter 0.

NumSess= **<u>1</u>** **|** ***number_of_sessions***
The number of sessions to allocate. The default is 1.

NumCmds= **<u>1</u>** **|** ***number_of_commands***
The number of commands to allocate. The default is 1.

NumNames= **<u>1</u>** **|** ***number_of_names***
The number of names to allocate. The default is 1.

Library1= ***DLLName1***
The name of the NetBIOS DLL.

The default value is NETAPI32.

Windows SPX (IBM MQ for Windows only)

Use the SPX queue manager properties page from the IBM MQ Explorer, or the SPX stanza in the qm.ini file, to specify SPX protocol configuration parameters.

Socket= **<u>5E86</u>** **|** ***socket_number***
The SPX socket number in hexadecimal notation. The default is X'5E86'.

BoardNum= **<u>0</u>** **|** ***adapter_number***
The LAN adapter number. The default is adapter 0.

KeepAlive=NO|YES
Switch the KeepAlive function on or off.

KeepAlive=YES causes SPX to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

Library1= ***DLLName1***
The name of the SPX DLL.

The default is WSOCK32.DLL.

Library2= ***DLLName2***
The same as LibraryName1, used if the code is stored in two separate libraries.

The default is WSOCK32.DLL.

ListenerBacklog=number
Override the default number of outstanding requests for the SPX listener.

When receiving on SPX, a maximum number of outstanding connection requests is set. This can be considered to be a backlog of requests waiting on the SPX socket for the listener to accept the request. The default listener backlog values are shown in Table 14 on page 117.

Table 14. Default outstanding connection requests (SPX)

Platform	Default ListenerBacklog value
Windows Server	100
Windows Workstation	5

Note: Some operating systems support a larger value than the default shown. Use this to avoid reaching the connection limit.

Conversely, some operating systems might limit the size of the SPX backlog, so the effective SPX backlog could be smaller than requested here.

If the backlog reaches the values shown in Table 14 on page 117, the SPX connection is rejected and the channel cannot start. For message channels, this results in the channel going into a RETRY state and retrying the connection at a later time. For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from `MQCONN` and should retry the connection at a later time.

Exit path

Use the `Exits` queue manager properties page from the IBM MQ Explorer, or the `ExitPath` stanza in the qm.ini file to specify the path for user exit programs on the queue manager system.

ExitsDefaultPath= *string*
The ExitsDefaultPath attribute specifies the location of:

- 32-bit channel exits for clients
- 32-bit channel exits and data conversion exits for servers
- Unqualified XA switch load files

ExitsDefaultPath64= *string*
The ExitsDefaultPath64 attribute specifies the location of:

- 64-bit channel exits for clients
- 64-bit channel exits and data conversion exits for servers
- Unqualified XA switch load files

API exits

For a server, use the `Exits` queue manager properties page from the MQ Explorer, or the `ApiExitLocal` stanza in the `qm.ini` file to identify API exit routines for a queue manager. For a client modify the `ApiExitLocal` stanza in the mqclient.ini file to identify API exit routines for a queue manager.

On Windows systems, you can also use the amqmdain command to change the entries for API exits. (To identify API exit routines for all queue managers, you use the `ApiExitCommon` and `ApiExitTemplate` stanzas, as described in "API exits" on page 102.)

Note, that for the API exit to work correctly, the message from the server must be sent to the client unconverted. After the API exit has processed the message, the message must then be converted on the client. This, therefore, requires that you have installed all conversion exits on the client.

For a complete description of the attributes for these stanzas, see Configuring API exits.

QMErrorLog stanza on UNIX, Linux, and Windows

Use the `Extended` queue manager properties page from the MQ Explorer, or the `QMErrorLog` stanza in the qm.ini file to tailor the operation and contents of queue manager error logs.

Attention: You can use MQ Explorer to make the changes, only if you are using a local queue manager on the Windows platform.

ErrorLogSize= *maxsize*
Specifies the size of the queue manager error log at which it is copied to the backup. *maxsize* must be in the range 32768 through 2147483648 bytes. If `ErrorLogSize` is not specified, the default value of 2097152 bytes (2 MB) is used.

ExcludeMessage= *msgIds*
Specifies messages that are not to be written to the queue manager error log. If your IBM MQ system is heavily used, with many channels stopping and starting, a large number of information messages are sent to the z/OS console and hardcopy log. The IBM MQ - IMS bridge and buffer manager might also produce a large number of information messages, so excluding messages prevents you from receiving a large number of messages if you require it. *msqIds* contain a comma-separated list of message id's from the following:

5211 - Maximum property name length exceeded.
5973 - Distributed publish/subscribe subscription inhibited
5974 - Distributed publish/subscribe publication inhibited
6254 - The system could not dynamically load shared library
IBM i 7163 - Job started message (IBM i only)
7234 - Number of messages loaded
9001 - Channel program ended normally
9002 - Channel program started
9202 - Remote host not available
9208 - Error on receive from host
9209 - Connection closed
9228 - Cannot start channel responder
9489 - SVRCONN max instances limit exceeded
9490 - SVRCONN max instances per client limit exceeded
9508 - Cannot connect to queue manager
9524 - Remote queue manager unavailable
9528 - User requested closure of channel
9545 - Disconnect interval expired
9558 - Remote Channel is not available
9637 - Channel is lacking a certificate
9776 - Channel was blocked by user ID
9777 - Channel was blocked by NOACCESS map
9782 - Connection was blocked by address
9999 - Channel program ended abnormally

SuppressMessage= *msgIds*
Specifies messages that are written to the queue manager error log once only in a specified time interval. If your IBM MQ system is heavily used, with many channels stopping and starting, a large number of information messages are sent to the z/OS console and hardcopy log. The IBM MQ - IMS bridge and buffer manager might also produce a large number of information messages, so suppressing messages prevents you from receiving a number of repeating messages if you require it. The time interval is specified by `SuppressInterval` . *msqIds* contain a comma-separated list of message id's from the following:

5211 - Maximum property name length exceeded.
5973 - Distributed publish/subscribe subscription inhibited

5974 - Distributed publish/subscribe publication inhibited
6254 - The system could not dynamically load shared library
IBM i 7163 - Job started message (IBM i only)
7234 - Number of messages loaded
9001 - Channel program ended normally
9002 - Channel program started
9202 - Remote host not available
9208 - Error on receive from host
9209 - Connection closed
9228 - Cannot start channel responder
9489 - SVRCONN max instances limit exceeded
9490 - SVRCONN max instances per client limit exceeded
9508 - Cannot connect to queue manager
9524 - Remote queue manager unavailable
9528 - User requested closure of channel
9545 - Disconnect interval expired
9558 - Remote Channel is not available
9637 - Channel is lacking a certificate
9776 - Channel was blocked by user ID
9777 - Channel was blocked by NOACCESS map
9782 - Connection was blocked by address
9999 - Channel program ended abnormally

If the same message id is specified in both `SuppressMessage` and `ExcludeMessage` , the message is excluded.

SuppressInterval= ***length***
Specifies the time interval, in seconds, in which messages specified in `SuppressMessage` are written to the queue manager error log once only. *length* must be in the range 1 through 86400 seconds. If `SuppressInterval` is not specified, the default value of 30 seconds is used.

Queue manager default bind type

Use the `Extended` queue manager properties page from the IBM MQ Explorer, or the `Connection` stanza in the qm.ini file to specify the default bind type. Note that you must create a `Connection` stanza if you need one.

DefaultBindType= **SHARED** **| ISOLATED**
If DefaultBindType is set to ISOLATED, applications and the queue manager run in separate processes, and no resources are shared between them.

If DefaultBindType is set to SHARED, applications and the queue manager run in separate processes, but some resources are shared between them.

The default is SHARED.

Attention: `DefaultBindType` applies to all MQCONN calls, and any using MQCONNX with MQCNO_STANDARD_BINDING.

Changing the **`DefaultBindType`** might cause some applications to degrade in performance.

SSL and TLS stanza of the queue manager configuration file

Use the SSL stanza of the queue manager configuration file to configure SSL or TLS channels on your queue manager.

Online Certificate Status Protocol (OCSP)

A certificate can contain an AuthorityInfoAccess extension. This extension specifies a server to be contacted through Online Certificate Status Protocol (OCSP). To allow SSL or TLS channels on your queue manager to use AuthorityInfoAccess extensions, ensure that the OCSP server named in them is available, is correctly configured, and is accessible over the network. For more information, see Working with revoked certificates.

CrlDistributionPoint (CDP)

A certificate can contain a CrlDistributionPoint extension. This extension contains a URL which identifies both the protocol used to download a certificate revocation list (CRL) and also the server to be contacted.

If you want to allow SSL or TLS channels on your queue manager to use CrlDistributionPoint extensions, ensure that the CDP server named in them is available, correctly configured, and accessible over the network.

The SSL Stanza

Use the SSL stanza in the `qm.ini` file to configure how SSL or TLS channels on your queue manager attempts to use the following facilities, and how they react if problems occur when using them.

In each of the following cases, if the value supplied is not one of the valid values listed, then the default value is taken. No error messages are written mentioning that an invalid value is specified.

CDPCheckExtensions= YES | NO

CDPCheckExtensions specifies whether SSL or TLS channels on this queue manager try to check CDP servers that are named in CrlDistributionPoint certificate extensions.

- YES: SSL or TLS channels try to check CDP servers to determine whether a digital certificate is revoked.
- `NO`: SSL or TLS channels do not try to check CDP servers. This value is the default.

OCSPAuthentication= REQUIRED | WARN | OPTIONAL

OCSPAuthentication specifies the action to be taken when a revocation status cannot be determined from an OCSP server.

If OCSP checking is enabled, an SSL or TLS channel program attempts to contact an OCSP server.

If the channel program is unable to contact any OCSP servers, or if no server can provide the revocation status of the certificate, then the value of the OCSPAuthentication parameter is used.

- `REQUIRED`: Failure to determine the revocation status causes the connection to be closed with an error. This value is the default.
- `WARN`: Failure to determine the revocation status causes a warning message to be written in the queue manager error log, but the connection is allowed to proceed.
- `OPTIONAL`: Failure to determine the revocation status allows the connection to proceed silently. No warnings or errors are given.

OCSPCheckExtensions= YES | NO

OCSPCheckExtensions specifies whether SSL and TLS channels on this queue manager try to check OCSP servers that are named in AuthorityInfoAccess certificate extensions.

- YES: SSL and TLS channels try to check OCSP servers to determine whether a digital certificate is revoked. This value is the default.

- NO: SSL and TLS channels do not try to check OCSP servers.

SSLHTTPProxyName= ***string***
: The string is either the host name or network address of the HTTP Proxy server that is to be used by GSKit for OCSP checks. This address can be followed by an optional port number, enclosed in parentheses. If you do not specify the port number, the default HTTP port, 80, is used. On the HP-UX PA-RISC and Sun Solaris SPARC platforms, and for 32-bit clients on AIX, the network address can only be an IPv4 address; on other platforms it can be an IPv4 or IPv6 address.

  This attribute might be necessary if, for example, a firewall prevents access to the URL of the OCSP responder.

Exit properties

Use the Cluster queue manager properties page from the MQ Explorer, or the ExitPropertiesLocal stanza in the qm.ini file, to specify information about exit properties on a queue manager. Alternatively, you can set it using the **amqmdain** command.

By default, this setting is inherited from the CLWLMode attribute in the ExitProperties stanza of the machine-wide configuration (described in "Exit properties" on page 98). Change this setting only if you want to configure this queue manager in a different way. This value can be overridden for individual queue managers using the cluster workload mode attribute on the Cluster queue manager properties page.

CLWLMode= SAFE |FAST
: The cluster workload (CLWL) exit allows you to specify which cluster queue in the cluster to open in response to an MQI call (for example, MQOPEN , MQPUT). The CLWL exit runs either in FAST mode or SAFE mode depending on the value you specify on the CLWLMode attribute. If you omit the CLWLMode attribute, the cluster workload exit runs in SAFE mode.

  SAFE
  : Run the CLWL exit in a separate process from the queue manager. This is the default.

    If a problem arises with the user-written CLWL exit when running in SAFE mode, the following happens:

    - The CLWL server process (amqzlwa0) fails.
    - The queue manager restarts the CLWL server process.
    - The error is reported to you in the error log. If an MQI call is in progress, you receive notification in the form of a return code.

    The integrity of the queue manager is preserved.

    Note: Running the CLWL exit in a separate process can affect performance.

  FAST
  : Run the cluster exit inline in the queue manager process.

    Specifying this option improves performance by avoiding the process switching costs associated with running in SAFE mode, but does so at the expense of queue manager integrity. You should only run the CLWL exit in FAST mode if you are convinced that there are **no** problems with your CLWL exit, and you are particularly concerned about performance.

    If a problem arises when the CLWL exit is running in FAST mode, the queue manager will fail and you run the risk of the integrity of the queue manager being compromised.

Subpool

This stanza is created by IBM MQ. Do not change it.

The stanza Subpool, and attribute ShortSubpoolName within that stanza, are written automatically by IBM MQ when you create a queue manager. IBM MQ chooses a value for ShortSubpoolName. Do not alter this value.

The name corresponds to a directory and symbolic link created inside the /var/mqm/sockets directory, which IBM MQ uses for internal communications between its running processes.

IBM i Linux UNIX Filesystem stanza of the qm.ini file

The default permissions set on the error log files are expected to be useful in most circumstances, and therefore there is no need for most IBM MQ administrators to alter them.

However, your IBM MQ administrator might want to alter the permissions on their error log files, in which case they should set the Filesystem stanza option **`ValidateAuth`**=*No*, which causes the queue manager to leave the permissions unaltered thereafter.

The default behavior (without **`ValidateAuth`**=*No*) is that the queue manager checks the file permissions of the queue manager error logs, and changes them back to their default values. This check can happen any time, including during a queue manager end or start operation.

Configuring distributed queuing

This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Before reading this section it is helpful to have an understanding of channels, queues, and the other concepts introduced in Distributed queuing and clusters.

Use the information in the following links to connect your applications using distributed queuing:

- "How to send a message to another queue manager" on page 146
- "Triggering channels" on page 166
- "Safety of messages" on page 164
- "IBM MQ distributed queuing techniques" on page 123
- "Introduction to distributed queue management" on page 143
- Windows Linux UNIX "Monitoring and controlling channels on Windows, UNIX and Linux platforms" on page 173
- IBM i "Monitoring and controlling channels on IBM i" on page 195

Related concepts

"Customizing IBM MQ for z/OS" on page 526
Use this topic as a step by step guide for customizing your IBM MQ system.

"Configuring connections between the client and server" on page 14
To configure the communication links between IBM MQ MQI clients and servers, decide on your communication protocol, define the connections at both ends of the link, start a listener, and define channels.

"Changing IBM MQ and queue manager configuration information" on page 80
Change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation.

"Configuring queue managers on z/OS" on page 522
Use these instructions to configure queue managers on IBM MQ for z/OS.

"Setting up communications with other queue managers" on page 587
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

Related tasks

"Configuring a queue manager cluster" on page 215

Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. You can define cluster components, and create and manage clusters.

IBM MQ distributed queuing techniques

The subtopics in this section describe techniques that are of use when planning channels. These subtopics describe techniques to help you plan how to connect your queue managers together, and manage the flow of messages between your applications.

For message channel planning examples, see:

- Message channel planning example for distributed platforms
- IBM i Message channel planning example for IBM MQ for IBM i
- z/OS Message channel planning example for z/OS
- z/OS Message channel planning example for z/OS using queue-sharing groups

Related concepts

"Configuring distributed queuing" on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Channels
Introduction to message queuing
Distributed queuing and clusters

Related reference

Example configuration information

Message flow control

Message flow control is a task that involves the setting up and maintenance of message routes between queue managers. It is important for routes that multi-hop through many queue managers. This section describes how you use queues, alias queue definitions, and message channels on your system to achieve message flow control.

You control message flow using a number of techniques that were introduced in "Configuring distributed queuing" on page 122. If your queue manager is in a cluster, message flow is controlled using different techniques, as described in "Message flow control" on page 123. z/OS If your queue managers are in a queue sharing group and intra-group queuing (IGQ) is enabled, then the message flow can be controlled by IGQ agents. These agents are described in Intra-group queuing.

You can use the following objects to achieve message flow control:

- Transmission queues
- Message channels
- Remote queue definition
- Queue manager alias definition
- Reply-to queue alias definition

The queue manager and queue objects are described in Object types. Message channels are described in Distributed queuing components. The following techniques use these objects to create message flows in your system:

- Putting messages to remote queues
- Routing by way of particular transmission queues
- Receiving messages
- Passing messages through your system

- Separating message flows
- Switching a message flow to another destination
- Resolving the reply-to queue name to an alias name

Note

All the concepts described in this section are relevant for all nodes in a network, and include sending and receiving ends of message channels. For this reason, only one node is illustrated in most examples. The exception is where the example requires explicit cooperation by the administrator at the other end of a message channel.

Before proceeding to the individual techniques, it is useful to recap on the concepts of name resolution and the three ways of using remote queue definitions. See Distributed queuing and clusters.

Related concepts

"Queue names in transmission header" on page 124
Destination queue names travel with the message in the transmission header until the destination queue has been reached.

"How to create queue manager and reply-to aliases" on page 124
This topic explains the three ways that you can create a remote queue definition.

Queue names in transmission header

Destination queue names travel with the message in the transmission header until the destination queue has been reached.

The queue name used by the application, the logical queue name, is resolved by the queue manager to the destination queue name. In other words, the physical queue name. This destination queue name travels with the message in a separate data area, the transmission header, until the destination queue has been reached. The transmission header is then stripped off.

You change the queue manager part of this queue name when you create parallel classes of service. Remember to return the queue manager name to the original name when the end of the class-of-service diversion has been reached.

How to create queue manager and reply-to aliases

This topic explains the three ways that you can create a remote queue definition.

The remote queue definition object is used in three different ways. Table 15 on page 125 explains how to define each of the three ways:

- Using a remote queue definition to redefine a local queue name.

  The application provides only the queue name when opening a queue, and this queue name is the name of the remote queue definition.

  The remote queue definition contains the names of the target queue and queue manager. Optionally, the definition can contain the name of the transmission queue to be used. If no transmission queue name is provided, the queue manager uses the queue manager name, taken from the remote queue definition, for the transmission queue name. If a transmission queue of this name is not defined, but a default transmission queue is defined, the default transmission queue is used.
- Using a remote queue definition to redefine a queue manager name.

  The application, or channel program, provides a queue name together with the remote queue manager name when opening the queue.

  If you have provided a remote queue definition with the same name as the queue manager name, and you have left the queue name in the definition blank, then the queue manager substitutes the queue manager name in the open call with the queue manager name in the definition.

In addition, the definition can contain the name of the transmission queue to be used. If no transmission queue name is provided, the queue manager takes the queue manager name, taken from the remote queue definition, for the transmission queue name. If a transmission queue of this name is not defined, but a default transmission queue is defined, the default transmission queue is used.

- Using a remote queue definition to redefine a reply-to queue name.

  Each time an application puts a message to a queue, it can provide the name of a reply-to queue for answer messages but with the queue manager name blank.

  If you provide a remote queue definition with the same name as the reply-to queue then the local queue manager replaces the reply-to queue name with the queue name from your definition.

  You can provide a queue manager name in the definition, but not a transmission queue name.

Table 15. Three ways of using the remote queue definition object

Usage	**Queue manager name**	**Queue name**	**Transmission queue name**
1. Remote queue definition (on OPEN call)			
Supplied in the call	blank or local QM	(*) required	not applicable
Supplied in the definition	required	required	optional
2. Queue manager alias (on OPEN call)			
Supplied in the call	(*) required and not local QM	required	not applicable
Supplied in the definition	required	blank	optional
3. Reply-to queue alias (on PUT call)			
Supplied in the call	blank	(*) required	not applicable
Supplied in the definition	optional	optional	blank

Note: (*) means that this name is the name of the definition object

For a formal description, see Queue name resolution.

Putting messages on remote queues

You can use remote queue definition objects to resolve a queue name to a transmission queue to an adjacent queue manager.

In a distributed-queuing environment, a transmission queue and channel are the focal point for all messages to a location whether the messages originate from applications in your local system, or arrive through channels from an adjacent system. Figure 6 on page 126 shows an application placing messages on a logical queue named 'QA_norm'. The name resolution uses the remote queue definition 'QA_norm' to select the transmission queue QMB. It then adds a transmission header to the messages stating 'QA_norm at QMB'.

Messages arriving from the adjacent system on 'Channel_back' have a transmission header with the physical queue name 'QA_norm at QMB', for example. These messages are placed unchanged on transmission queue QMB.

The channel moves the messages to an adjacent queue manager.

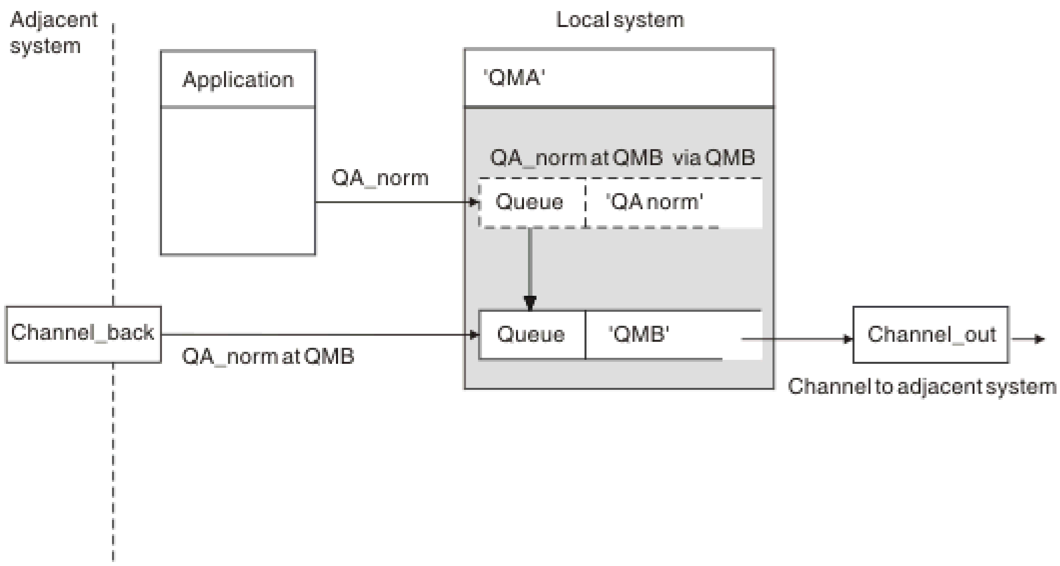

Figure 6. A remote queue definition is used to resolve a queue name to a transmission queue to an adjacent queue manager

If you are the IBM MQ system administrator, you must:

- Define the message channel from the adjacent system
- Define the message channel to the adjacent system
- Create the transmission queue QMB
- Define the remote queue object 'QA_norm' to resolve the queue name used by applications to the destination queue name, destination queue manager name, and transmission queue name

In a clustering environment, you only need to define a cluster-receiver channel at the local queue manager. You do not need to define a transmission queue or a remote queue object. See Clusters.

More about name resolution

The effect of the remote queue definition is to define a physical destination queue name and queue manager name. These names are put in the transmission headers of messages.

Incoming messages from an adjacent system have already had this type of name resolution carried out by the original queue manager. Therefore they have the transmission header showing the physical destination queue name and queue manager name. These messages are unaffected by remote queue definitions.

Choosing the transmission queue

You can use a remote queue definition to allow a different transmission queue to send messages to the same adjacent queue manager.

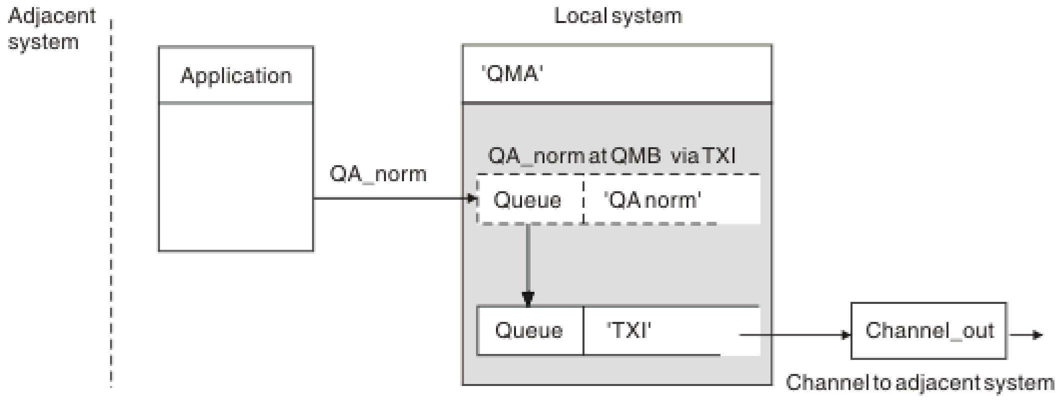

Figure 7. The remote queue definition allows a different transmission queue to be used

In a distributed-queuing environment, when you need to change a message flow from one channel to another, use the same system configuration as shown in Figure 6 on page 126 in "Putting messages on remote queues" on page 125. Figure 7 on page 127 in this topic shows how you use the remote queue definition to send messages over a different transmission queue, and therefore over a different channel, to the same adjacent queue manager.

For the configuration shown in Figure 7 on page 127, you must provide the remote queue object 'QA_norm', and the transmission queue 'TX1'. You must provide 'QA_norm' to choose the 'QA_norm' queue at the remote queue manager, the transmission queue 'TX1', and the queue manager 'QMB_priority'. Specify 'TX1' in the definition of the channel adjacent to the system.

Messages are placed on transmission queue 'TX1' with a transmission header containing 'QA_norm at QMB_priority', and are sent over the channel to the adjacent system.

The channel_back has been left out of this illustration because it would need a queue manager alias.

In a clustering environment, you do not need to define a transmission queue or a remote queue definition. For more information, see "Defining cluster queues" on page 216.

Receiving messages

You can configure the queue manager to receive messages from other queue managers. You must ensure that unintentional name resolution does not occur.

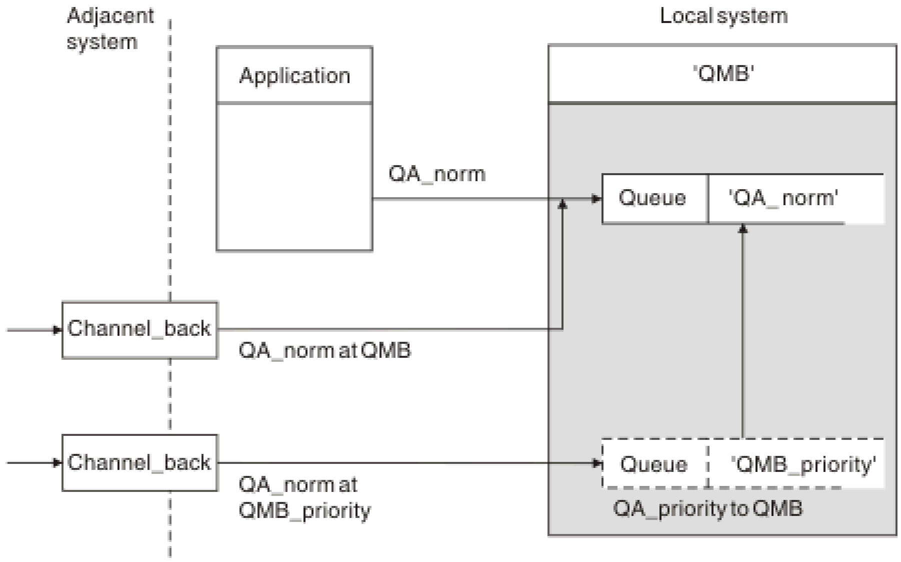

Figure 8. Receiving messages directly, and resolving alias queue manager name

As well as arranging for messages to be sent, the system administrator must also arrange for messages to be received from adjacent queue managers. Received messages contain the physical name of the destination queue manager and queue in the transmission header. They are treated the same as messages from a local application that specifies both queue manager name and queue name. Because of this treatment, you need to ensure that messages entering your system do not have an unintentional name resolution carried out. See Figure 8 on page 128 for this scenario.

For this configuration, you must prepare:

- Message channels to receive messages from adjacent queue managers
- A queue manager alias definition to resolve an incoming message flow, 'QMB_priority', to the local queue manager name, 'QMB'
- The local queue, 'QA_norm', if it does not exist

Receiving alias queue manager names

The use of the queue manager alias definition in this illustration has not selected a different destination queue manager. Messages passing through this local queue manager and addressed to 'QMB_priority' are intended for queue manager 'QMB'. The alias queue manager name is used to create the separate message flow.

Passing messages through your system

You can pass messages through your system in three ways - using the location name, using an alias for the queue manager, or selecting a transmission queue.

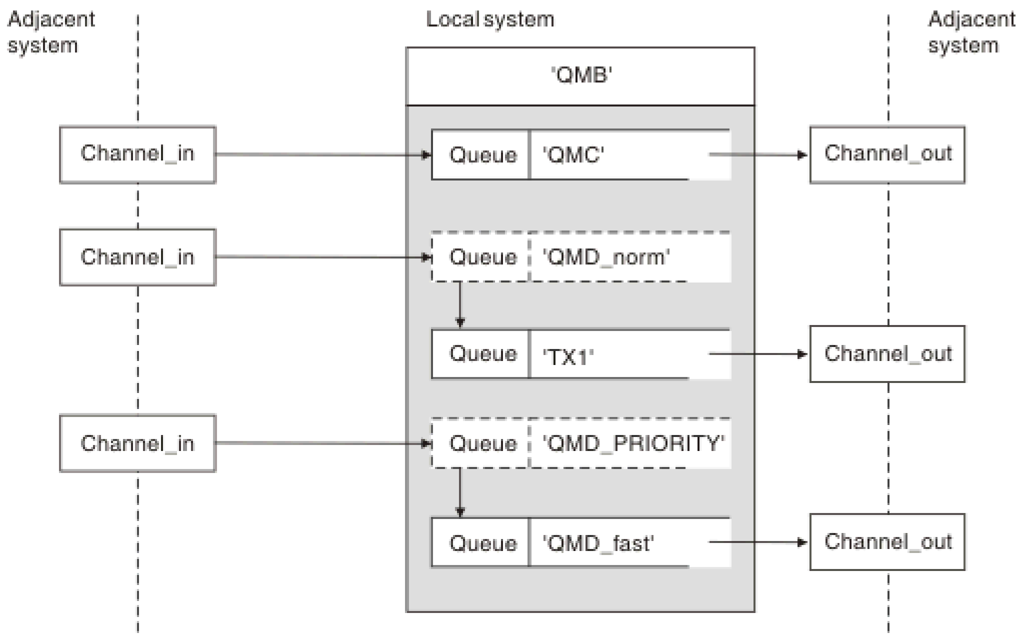

Figure 9. Three methods of passing messages through your system

The technique shown in Figure 8 on page 128 in "Receiving messages" on page 128, showed how an alias flow is captured. Figure 9 on page 129 illustrates the ways networks are built up by bringing together the techniques previously described.

The configuration shows a channel delivering three messages with different destinations:

1. QB at `QMC`
2. QB at `QMD_norm`
3. QB at `QMD_PRIORITY`

You must pass the first message flow through your system unchanged. You must pass the second message flow through a different transmission queue and channel. For the second message flow you must also resolve messages for the alias queue manager name `QMD_norm` to the queue manager `QMD`. The third message flow chooses a different transmission queue without any other change.

In a clustering environment, messages are passed through a cluster transmission queue. Normally a single transmission queue, `SYSTEM.CLUSTER.TRANSMIT.QUEUE`, transfers all messages to all queue managers in all clusters that the queue manager is a member of; see A cluster of queue managers. You can define separate transmission queues for all or some of the queue managers in the clusters that the queue manager is a member of.

The following methods describe techniques applicable to a distributed-queuing environment.

Use these methods

For these configurations, you must prepare the:

- Input channel definitions
- Output channel definitions
- Transmission queues:
  - `QMC`

  - TX1
  - QMD_fast
- Queue manager alias definitions:
  - QMD_norm with QMD_norm to QMD through TX1
  - QMD_PRIORITY with QMD_PRIORITY to QMD_PRIORITY through QMD_fast

Note: None of the message flows shown in the example changes the destination queue. The queue manager name aliases provide separation of message flows.

Method 1: Use the incoming location name

You are going to receive messages with a transmission header containing another location name, such as QMC. The simplest configuration is to create a transmission queue with that name, QMC. The channel that services the transmission queue delivers the message unchanged to the next destination.

Method 2: Use an alias for the queue manager

The second method is to use the queue manager alias object definition, but specify a new location name, QMD, and a particular transmission queue, TX1. This action:

- Terminates the alias message flow setup by the queue manager name alias QMD_norm, that is, the named class of service QMD_norm.
- Changes the transmission headers on these messages from QMD_norm to QMD.

Method 3: Select a transmission queue

The third method is to have a queue manager alias object defined with the same name as the destination location, QMD_PRIORITY. Use the queue manager alias definition to select a particular transmission queue, QMD_fast, and therefore another channel. The transmission headers on these messages remain unchanged.

Separating message flows

You can use a queue manager alias to create separate message flows to send messages to the same queue manager.

In a distributed-queuing environment, the need to separate messages to the same queue manager into different message flows can arise for a number of reasons. For example:

- You might need to provide a separate flow for large, medium, and small messages. This need also applies in a clustering environment and, in this case, you can create clusters that overlap. There are a number of reasons you might do so, for example:
  - To allow different organizations to have their own administration.
  - To allow independent applications to be administered separately.
  - To create a class of service. For example, you could have a cluster called STAFF that is a subset of the cluster called STUDENTS. When you put a message to a queue advertised in the STAFF cluster, a restricted channel is used. When you put a message to a queue advertised in the STUDENTS cluster, either a general channel or a restricted channel can be used.
  - To create test and production environments.
- It might be necessary to route incoming messages by different paths from the path of the locally generated messages.
- Your installation might require to schedule the movement of messages at certain times (for example, overnight) and the messages then need to be stored in reserved queues until scheduled.

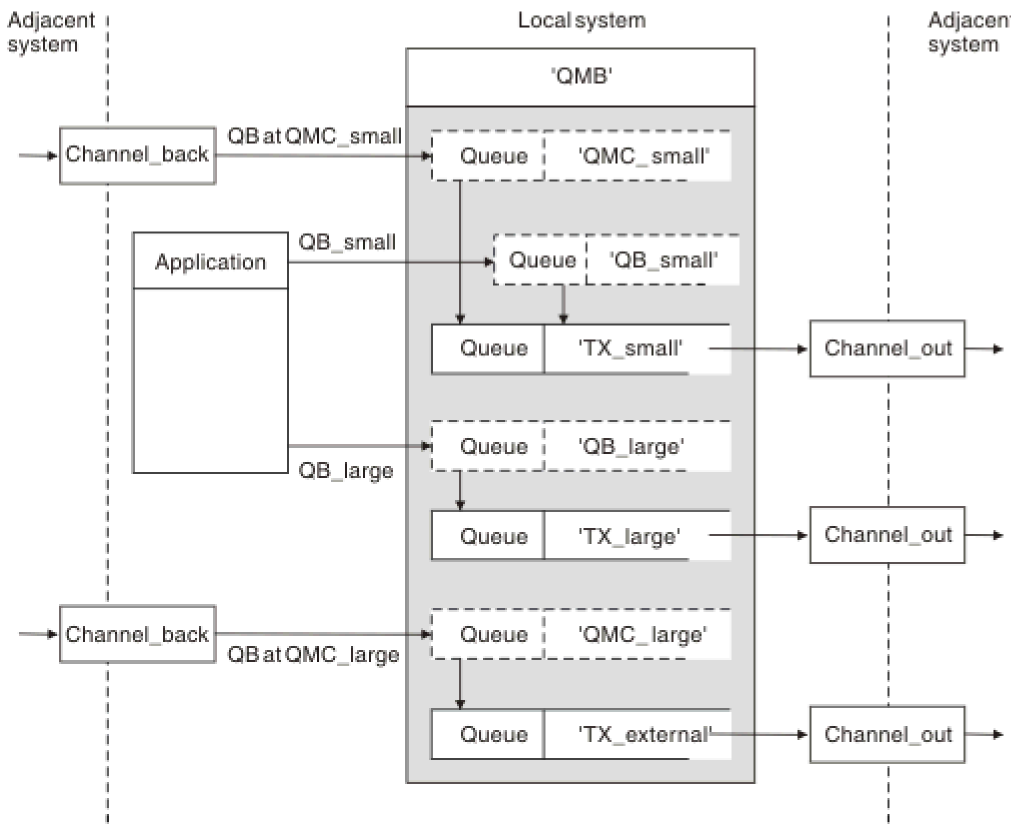

Figure 10. Separating messages flows

In the example shown in Figure 10 on page 131, the two incoming flows are to alias queue manager names 'QMC_small' and 'QMC_large'. You provide these flows with a queue manager alias definition to capture these flows for the local queue manager. You have an application addressing two remote queues and you need these message flows to be kept separate. You provide two remote queue definitions that specify the same location, 'QMC', but specify different transmission queues. This definition keeps the flows separate, and nothing extra is needed at the far end as they have the same destination queue manager name in the transmission headers. You provide:

- The incoming channel definitions
- The two remote queue definitions QB_small and QB_large
- The two queue manager alias definitions QMC_small and QMC_large
- The three sending channel definitions
- Three transmission queues: TX_small, TX_large, and TX_external

Coordination with adjacent systems

When you use a queue manager alias to create a separate message flow, you need to coordinate this activity with the system administrator at the remote end of the message channel to ensure that the corresponding queue manager alias is available there.

Concentrating messages to diverse locations

You can concentrate messages destined for various locations on to a single channel.

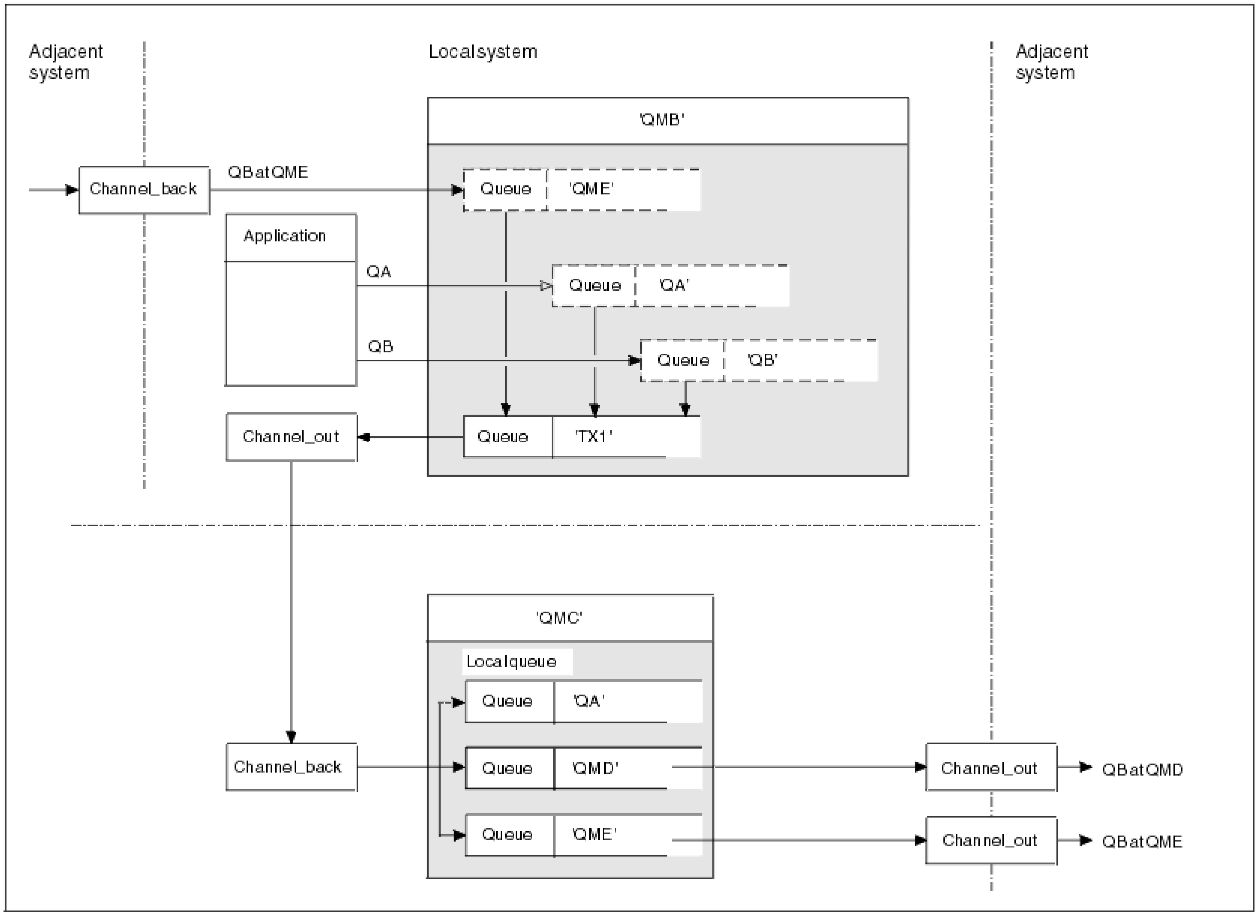

Figure 11. Combining message flows on to a channel

Figure 11 on page 132 illustrates a distributed-queuing technique for concentrating messages that are destined for various locations on to one channel. Two possible uses would be:

- Concentrating message traffic through a gateway
- Using wide bandwidth highways between nodes

In this example, messages from different sources, local and adjacent, and having different destination queues and queue managers, are flowed through transmission queue 'TX1' to queue manager QMC. Queue manager QMC delivers the messages according to the destinations. One set to a transmission queue 'QMD' for onward transmission to queue manager QMD. Another set to a transmission queue 'QME' for onward transmission to queue manager QME. Other messages are put on the local queue 'QA'.

You must provide:

- Channel definitions
- Transmission queue TX1
- Remote queue definitions:
  - QA with 'QA at QMC through TX1'
  - QB with 'QB at QMD through TX1'
- Queue manager alias definition:
  - QME with 'QME through TX1'

The complementary administrator who is configuring QMC must provide:

- Receiving channel definition with the same channel name
- Transmission queue QMD with associated sending channel definition
- Transmission queue QME with associated sending channel definition
- Local queue object QA.

Diverting message flows to another destination

You can redefine the destination of certain messages using queue manager aliases and transmission queues.

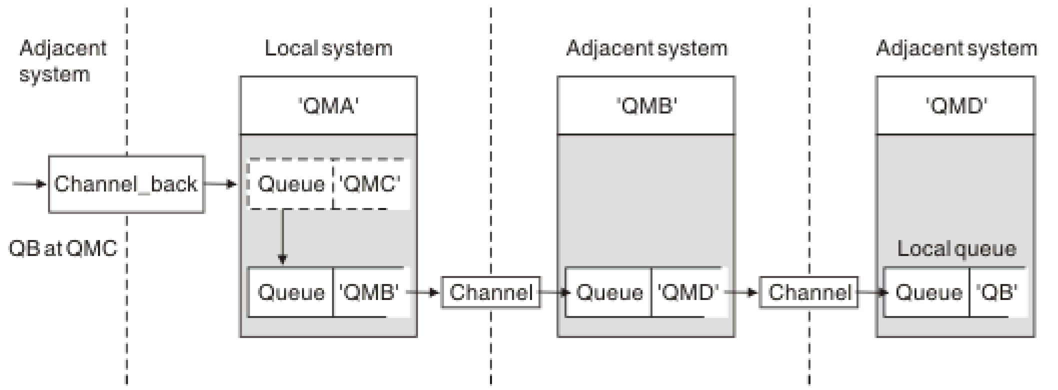

Figure 12. Diverting message streams to another destination

Figure 12 on page 133 illustrates how you can redefine the destination of certain messages. Incoming messages to QMA are destined for 'QB at QMC'. They normally arrive at QMA and be placed on a transmission queue called QMC which has been part of a channel to QMC. QMA must divert the messages to QMD, but is able to reach QMD only over QMB. This method is useful when you need to move a service from one location to another, and allow subscribers to continue to send messages on a temporary basis until they have adjusted to the new address.

The method of rerouting incoming messages destined for a certain queue manager to a different queue manager uses:

- A queue manager alias to change the destination queue manager to another queue manager, and to select a transmission queue to the adjacent system
- A transmission queue to serve the adjacent queue manager
- A transmission queue at the adjacent queue manager for onward routing to the destination queue manager

You must provide:

- Channel_back definition
- Queue manager alias object definition QMC with QB at QMD through QMB
- Channel_out definition
- The associated transmission queue QMB

The complementary administrator who is configuring QMB must provide:

- The corresponding channel_back definition
- The transmission queue, QMD
- The associated channel definition to QMD

You can use aliases within a clustering environment. For information, see “Queue-manager aliases and clusters” on page 307.

Sending messages to a distribution list

You can use a single MQPUT call to have an application send a message to several destinations.

In IBM MQ on all platforms except z/OS, an application can send a message to several destinations with a single MQPUT call. You can do so in both a distributed-queuing environment and a clustering environment. You have to define the destinations in a distribution list, as described in Distribution lists.

Not all queue managers support distribution lists. When an MCA establishes a connection with a partner, it determines whether the partner supports distribution lists and sets a flag on the transmission queue accordingly. If an application tries to send a message that is destined for a distribution list but the partner does not support distribution lists, the sending MCA intercepts the message and puts it onto the transmission queue once for each intended destination.

A receiving MCA ensures that messages sent to a distribution list are safely received at all the intended destinations. If any destinations fail, the MCA establishes which ones have failed. It then can generate exception reports for them and can try to send the messages to them again.

Reply-to queue

You can create a complete remote queue processing loop using a reply-to queue.

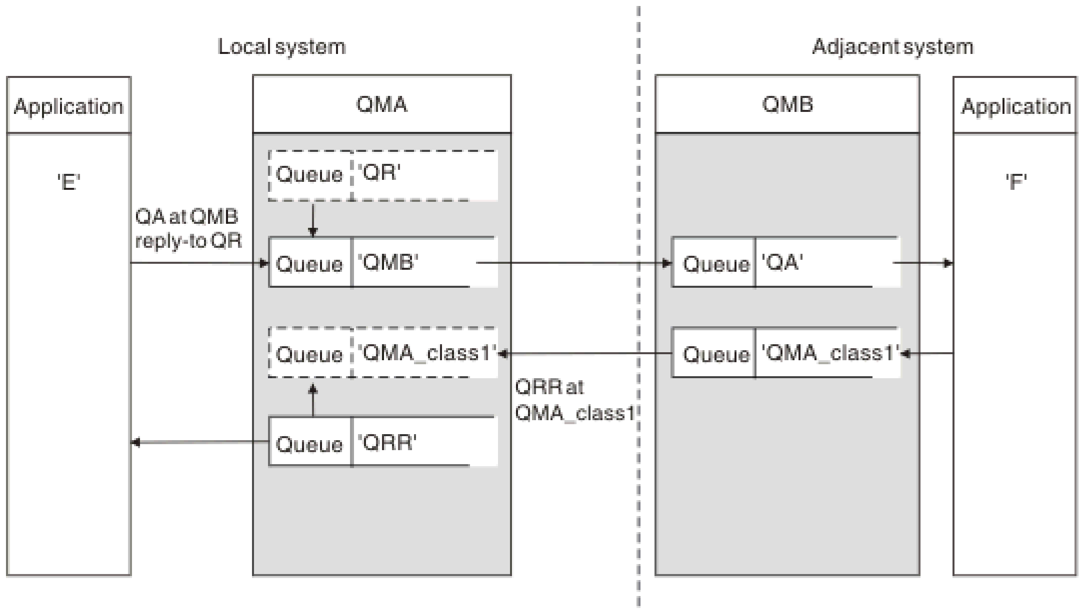

Figure 13. Reply-to queue name substitution during PUT call

A complete remote queue processing loop using a reply-to queue is shown in Figure 13 on page 134. This loop applies in both a distributed-queuing environment and a clustering environment. The details are as shown in Table 19 on page 141.

The application opens QA at QMB and puts messages on that queue. The messages are given a reply-to queue name of QR, without the queue manager name being specified. Queue manager QMA finds the reply-to queue object QR and extracts from it the alias name of QRR and the queue manager name QMA_class1. These names are put into the reply-to fields of the messages.

Reply messages from applications at QMB are addressed to QRR at QMA_class1. The queue manager alias name definition QMA_class1 is used by the queue manager to flow the messages to itself, and to queue QRR.

This scenario depicts the way you give applications the facility to choose a class of service for reply messages. The class is implemented by the transmission queue QMA_class1 at QMB, together with the queue manager alias definition, QMA_class1 at QMA. In this way, you can change an application's reply-to queue so that the flows are segregated without involving the application. The application always chooses

QR for this particular class of service. You have the opportunity to change the class of service with the reply-to queue definition QR.

You must create:

- Reply-to queue definition QR
- Transmission queue object QMB
- Channel_out definition
- Channel_back definition
- Queue manager alias definition QMA_class1
- Local queue object QRR, if it does not exist

The complementary administrator at the adjacent system must create:

- Receiving channel definition
- Transmission queue object QMA_class1
- Associated sending channel
- Local queue object QA.

Your application programs use:

- Reply-to queue name QR in put calls
- Queue name QRR in get calls

In this way, you can change the class of service as necessary, without involving the application. You change the reply-to alias 'QR', together with the transmission queue 'QMA_class1' and queue manager alias 'QMA_class1'.

If no reply-to alias object is found when the message is put on the queue, the local queue manager name is inserted in the blank reply-to queue manager name field. The reply-to queue name remains unchanged.

Name resolution restriction

Because the name resolution has been carried out for the reply-to queue at 'QMA' when the original message was put, no further name resolution is allowed at 'QMB'. The message is put with the physical name of the reply-to queue by the replying application.

The applications must be aware that the name they use for the reply-to queue is different from the name of the actual queue where the return messages are to be found.

For example, when two classes of service are provided for the use of applications with reply-to queue alias names of 'C1_alias', and 'C2_alias', the applications use these names as reply-to queue names in the message put calls. However, the applications actually expect messages to appear in queues 'C1' for 'C1_alias' and 'C2' for 'C2_alias'.

However, an application is able to make an inquiry call on the reply-to alias queue to check for itself the name of the real queue it must use to get the reply messages.

Related concepts

"How to create queue manager and reply-to aliases" on page 124
This topic explains the three ways that you can create a remote queue definition.

"Reply-to queue alias example" on page 136
This example illustrates the use of a reply-to alias to select a different route (transmission queue) for returned messages. The use of this facility requires the reply-to queue name to be changed in cooperation with the applications.

"How the example works" on page 137
An explanation of the example and how the queue manager uses the reply-to queue alias.

"Reply-to queue alias walk-through" on page 138

A walk-through of the process from an application putting a message on a remote queue through to the same application removing the reply message from the alias reply-to queue.

Reply-to queue alias example

This example illustrates the use of a reply-to alias to select a different route (transmission queue) for returned messages. The use of this facility requires the reply-to queue name to be changed in cooperation with the applications.

As shown in Figure 14 on page 136, the return route must be available for the reply messages, including the transmission queue, channel, and queue manager alias.

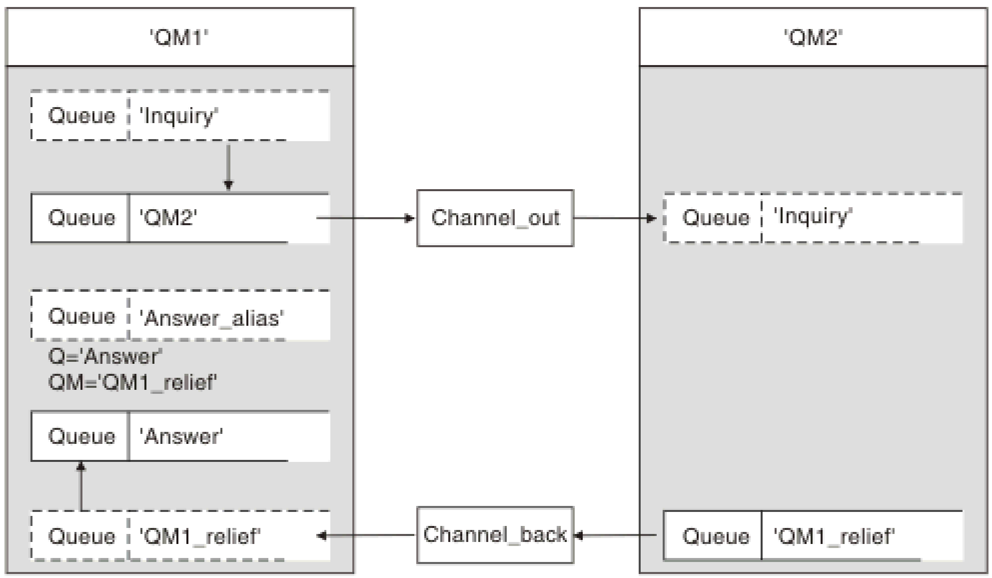

Figure 14. Reply-to queue alias example

This example is for requester applications at 'QM1' that send messages to server applications at 'QM2'. The messages on the server are to be returned through an alternative channel using transmission queue 'QM1_relief' (the default return channel would be served with a transmission queue 'QM1').

The reply-to queue alias is a particular use of the remote queue definition named 'Answer_alias'. Applications at QM1 include this name, 'Answer_alias', in the reply-to field of all messages that they put on queue 'Inquiry'.

Reply-to queue definition 'Answer_alias' is defined as 'Answer at QM1_relief'. Applications at QM1 expect their replies to appear in the local queue named 'Answer'.

Server applications at QM2 use the reply-to field of received messages to obtain the queue and queue manager names for the reply messages to the requester at QM1.

Definitions used in this example at QM1

The IBM MQ system administrator at QM1 must ensure that the reply-to queue 'Answer' is created along with the other objects. The name of the queue manager alias, marked with a '*', must agree with the queue manager name in the reply-to queue alias definition, also marked with an '*'.

Object	**Definition**	
Local transmission queue	QM2	
Remote queue definition	Object name	Inquiry
	Remote queue manager name	QM2
	Remote queue name	Inquiry
	Transmission queue name	QM2 (DEFAULT)

Object	Definition	
Queue manager alias	Object name	QM1_relief *
	Queue manager name	QM1
	Queue name	(blank)
Reply-to queue alias	Object name	Answer_alias
	Remote queue manager name	QM1_relief *
	Remote queue name	Answer

Put definition at QM1

Applications fill the reply-to fields with the reply-to queue alias name, and leave the queue manager name field blank.

Field	Content
Queue name	Inquiry
Queue manager name	(blank)
Reply-to queue name	Answer_alias
Reply-to queue manager	(blank)

Definitions used in this example at QM2

The IBM MQ system administrator at QM2 must ensure that the local queue exists for the incoming messages, and that the correctly named transmission queue is available for the reply messages.

Object	Definition
Local queue	Inquiry
Transmission queue	QM1_relief

Put definition at QM2

Applications at QM2 retrieve the reply-to queue name and queue manager name from the original message and use them when putting the reply message on the reply-to queue.

Field	Content
Queue name	Answer
Queue manager name	QM1_relief

How the example works

An explanation of the example and how the queue manager uses the reply-to queue alias.

In this example, requester applications at QM1 always use 'Answer_alias' as the reply-to queue in the relevant field of the put call. They always retrieve their messages from the queue named 'Answer'.

The reply-to queue alias definitions are available for use by the QM1 system administrator to change the name of the reply-to queue 'Answer', and of the return route 'QM1_relief'.

Changing the queue name 'Answer' is normally not useful because the QM1 applications are expecting their answers in this queue. However, the QM1 system administrator is able to change the return route (class of service), as necessary.

How the queue manager uses the reply-to queue alias

Queue manager QM1 retrieves the definitions from the reply-to queue alias when the reply-to queue name, included in the put call by the application, is the same as the reply-to queue alias, and the queue manager part is blank.

The queue manager replaces the reply-to queue name in the put call with the queue name from the definition. It replaces the blank queue manager name in the put call with the queue manager name from the definition.

These names are carried with the message in the message descriptor.

Table 16. Reply-to queue alias

Field name	Put call	Transmission header
Reply-to queue name	Answer_alias	Answer
Reply-to queue manager name	(blank)	QM1_relief

Reply-to queue alias walk-through

A walk-through of the process from an application putting a message on a remote queue through to the same application removing the reply message from the alias reply-to queue.

To complete this example, let us look at the process.

1. The application opens a queue named 'Inquiry', and puts messages to it. The application sets the reply-to fields of the message descriptor to:

Reply-to queue name	**Answer_alias**
Reply-to queue manager name	(blank)

2. Queue manager 'QM1' responds to the blank queue manager name by checking for a remote queue definition with the name 'Answer_alias'. If none is found, the queue manager places its own name, 'QM1', in the reply-to queue manager field of the message descriptor.
3. If the queue manager finds a remote queue definition with the name 'Answer_alias', it extracts the queue name and queue manager names from the definition (queue name='Answer' and queue manager name= 'QM1_relief'). It then puts them into the reply-to fields of the message descriptor.
4. The queue manager 'QM1' uses the remote queue definition 'Inquiry' to determine that the intended destination queue is at queue manager 'QM2', and the message is placed on the transmission queue 'QM2'. 'QM2' is the default transmission queue name for messages destined for queues at queue manager 'QM2'.
5. When queue manager 'QM1' puts the message on the transmission queue, it adds a transmission header to the message. This header contains the name of the destination queue, 'Inquiry', and the destination queue manager, 'QM2'.
6. The message arrives at queue manager 'QM2', and is placed on the 'Inquiry' local queue.
7. An application gets the message from this queue and processes the message. The application prepares a reply message, and puts this reply message on the reply-to queue name from the message descriptor of the original message:

Reply-to queue name	**Answer**
Reply-to queue manager name	QM1_relief

8. Queue manager 'QM2' carries out the put command. Finding that the queue manager name, 'QM1_relief', is a remote queue manager, it places the message on the transmission queue with the same name, 'QM1_relief'. The message is given a transmission header containing the name of the destination queue, 'Answer', and the destination queue manager, 'QM1_relief'.

9. The message is transferred to queue manager 'QM1'. The queue manager, recognizes that the queue manager name 'QM1_relief' is an alias, extracts from the alias definition 'QM1_relief' the physical queue manager name 'QM1'.
10. Queue manager 'QM1' then puts the message on the queue name contained in the transmission header, 'Answer'.
11. The application extracts its reply message from the queue 'Answer'.

Networking considerations

In a distributed-queuing environment, because message destinations are addressed with just a queue name and a queue manager name, certain rules apply.

1. Where the queue manager name is given, and the name is different from the local queue manager name:
   - A transmission queue must be available with the same name. This transmission queue must be part of a message channel moving messages to another queue manager, or
   - A queue manager alias definition must exist to resolve the queue manager name to the same, or another queue manager name, and optional transmission queue, or
   - If the transmission queue name cannot be resolved, and a default transmission queue has been defined, the default transmission queue is used.
2. Where only the queue name is supplied, a queue of any type but with the same name must be available on the local queue manager. This queue can be a remote queue definition which resolves to: a transmission queue to an adjacent queue manager, a queue manager name, and an optional transmission queue.

To see how this works in a clustering environment, see Clusters.

z/OS If the queue managers are running in a queue-sharing group (QSG) and intra-group queuing (IGQ) is enabled, you can use the SYSTEM.QSG.TRANSMIT.QUEUE. For more information, see Intra-group queuing.

Consider the scenario of a message channel moving messages from one queue manager to another in a distributed-queuing environment.

The messages being moved have originated from any other queue manager in the network, and some messages might arrive that have an unknown queue manager name as destination. This issue can occur when a queue manager name has changed or has been removed from the system, for example.

The channel program recognizes this situation when it cannot find a transmission queue for these messages, and places the messages on your undelivered-message (dead-letter) queue. It is your responsibility to look for these messages and arrange for them to be forwarded to the correct destination. Alternatively, return them to the originator, where the originator can be ascertained.

Exception reports are generated in these circumstances, if report messages were requested in the original message.

Name resolution convention

Name resolution that changes the identity of the destination queue (that is, logical to physical name changing), only occurs once, and only at the originating queue manager.

Subsequent use of the various alias possibilities must only be used when separating and combining message flows.

Return routing

Messages can contain a return address in the form of the name of a queue and queue manager. This return address form can be used in both a distributed-queuing environment and a clustering environment.

This address is normally specified by the application that creates the message. It can be modified by any application that then handles the message, including user exit applications.

Irrespective of the source of this address, any application handling the message might choose to use this address for returning answer, status, or report messages to the originating application.

The way these response messages are routed is not different from the way the original message is routed. You need to be aware that the message flows you create to other queue managers need corresponding return flows.

Physical name conflicts

The destination reply-to queue name has been resolved to a physical queue name at the original queue manager. It must not be resolved again at the responding queue manager.

It is a likely possibility for name conflict problems that can only be prevented by a network-wide agreement on physical and logical queue names.

Managing queue name translations

When you create a queue manager alias definition or a remote queue definition, the name resolution is carried out for every message carrying that name. This situation must be managed.

This description is provided for application designers and channel planners concerned with an individual system that has message channels to adjacent systems. It takes a local view of channel planning and control.

When you create a queue manager alias definition or a remote queue definition, the name resolution is carried out for every message carrying that name, regardless of the source of the message. To oversee this situation, which might involve large numbers of queues in a queue manager network, you keep tables of:

- The names of source queues and of source queue managers with respect to resolved queue names, resolved queue manager names, and resolved transmission queue names, with method of resolution
- The names of source queues with respect to:
  - Resolved destination queue names
  - Resolved destination queue manager names
  - Transmission queues
  - Message channel names
  - Adjacent system names
  - Reply-to queue names

Note: The use of the term *source* in this context refers to the queue name or the queue manager name provided by the application, or a channel program when opening a queue for putting messages.

An example of each of these tables is shown in Table 17 on page 141, Table 18 on page 141, and Table 19 on page 141.

The names in these tables are derived from the examples in this section, and this table is not intended as a practical example of queue name resolution in one node.

Table 17. Queue name resolution at queue manager QMA					
Source queue specified when queue is opened	**Source queue manager specified when queue is opened**	**Resolved queue name**	**Resolved queue manager name**	**Resolved transmission queue name**	**Resolution type**
QA_norm	-	QA_norm	QMB	QMB	Remote queue
(any)	QMB	-	-	QMB	(none)
QA_norm	-	QA_norm	QMB	TX1	Remote queue
QB	QMC	QB	QMD	QMB	Queue manager alias

Table 18. Queue name resolution at queue manager QMB					
Source queue specified when queue is opened	**Source queue manager specified when queue is opened**	**Resolved queue name**	**Resolved queue manager name**	**Resolved transmission queue name**	**Resolution type**
QA_norm	-	QA_norm	QMB	-	(none)
QA_norm	QMB	QA_norm	QMB	-	(none)
QA_norm	QMB_PRIORITY	QA_norm	QMB	-	Queue manager alias
(any)	QMC	(any)	QMC	QMC	(none)
(any)	QMD_norm	(any)	QMD_norm	TX1	Queue manager alias
(any)	QMD_PRIORITY	(any)	QMD_PRIORITY	QMD_fast	Queue manager alias
(any)	QMC_small	(any)	QMC_small	TX_small	Queue manager alias
(any)	QMC_large	(any)	QMC_large	TX_external	Queue manager alias
QB_small	QMC	QB_small	QMC	TX_small	Remote queue
QB_large	QMC	QB_large	QMC	TX_large	Remote queue
(any)	QME	(any)	QME	TX1	Queue manager alias
QA	QMC	QA	QMC	TX1	Remote queue
QB	QMD	QB	QMD	TX1	Remote queue

Table 19. Reply-to queue name translation at queue manager QMA			
Application design		**Reply-to alias definition**	
Local QMGR	**Queue name for messages**	**Reply-to queue alias name**	**Redefined to**
QMA	QRR	QR	QRR at QMA_class1

Channel message sequence numbering

The channel uses sequence numbers to check that messages are delivered in the same order as they are taken from the transmission queue.

Channel sequence numbers are checked when a channel is started and should a mismatch occur, it implies that persistent synchronization data has been lost on either side of the channel; for example, a disaster recovery (DR) configuration, or that end of batch processing was interrupted when the channel was in-doubt.

Issuing a RESET CHANNEL command does not cause loss or duplication of messages. The RESET acknowledges the warning from IBM MQ that something does not appear to be right. An indoubt channel that has lost persistent state continues to fail to startup after a RESET until you issue a RESOLVE CHANNEL command; it is that action that has the potential to lose or duplicate a batch.

This information can be displayed using DISPLAY CHSTATUS. The sequence number and an identifier called the LUWID are stored in persistent storage for the last message transferred in a batch. These values are used during channel start-up to ensure that both ends of the link agree on which messages have been transferred successfully.

Sequential retrieval of messages

If an application puts a sequence of messages to the same destination queue, those messages can be retrieved in sequence by a ***single*** application with a sequence of MQGET operations, if the following conditions are met:

- All the put requests were done from the same application.
- All the put requests were either from the same unit of work, or all the put requests were made outside of a unit of work.
- The messages all have the same priority.
- The messages all have the same persistence.
- For remote queuing, the configuration is such that there can only be one path from the application making the put request, through its queue manager, through intercommunication, to the destination queue manager and the target queue.
- The messages are not put to a dead-letter queue (for example, if a queue is temporarily full).
- The application getting the message does not deliberately change the order of retrieval, for example by specifying a particular *MsgId* or *CorrelId* or by using message priorities.
- Only one application is doing get operations to retrieve the messages from the destination queue. If there is more than one application, these applications must be designed to get all the messages in each sequence put by a sending application.

Note: Messages from other tasks and units of work might be interspersed with the sequence, even where the sequence was put from within a single unit of work.

If these conditions cannot be met, and the order of messages on the target queue is important, then the application can be coded to use its own message sequence number as part of the message to assure the order of the messages.

Sequence of retrieval of fast, nonpersistent messages

Nonpersistent messages on a fast channel might overtake persistent messages on the same channel and so arrive out of sequence. The receiving MCA puts the nonpersistent messages on the destination queue immediately and makes them visible. Persistent messages are not made visible until the next sync point.

Loopback testing

Loopback testing is a technique on non- z/OS platforms that allows you to test a communications link without actually linking to another machine.

You set up a connection between two queue managers as though they are on separate machines, but you test the connection by looping back to another process on the same machine. This technique means that you can test your communications code without requiring an active network.

The way you do so depends on which products and protocols you are using.

On Windows systems, you can use the "loopback" adapter.

Refer to the documentation for the products you are using for more information.

Route tracing and activity recording

You can confirm the route a message takes through a series of queue managers in two ways.

You can use the IBM MQ display route application, available through the control command `dspmqrte` , or you can use activity recording. Both of these topics are described in Monitoring reference.

Introduction to distributed queue management

Distributed queue management (DQM) is used to define and control communication between queue managers.

Distributed queue management:

- Enables you to define and control communication channels between queue managers
- Provides you with a message channel service to move messages from a type of *local queue*, known as a transmission queue, to communication links on a local system, and from communication links to local queues at a destination queue manager
- Provides you with facilities for monitoring the operation of channels and diagnosing problems, using panels, commands, and programs

Channel definitions associate channel names with transmission queues, communication link identifiers, and channel attributes. Channel definitions are implemented in different ways on different platforms. Message sending and receiving is controlled by programs known as *message channel agents* (MCAs), which use the channel definitions to start up and control communication.

The MCAs in turn are controlled by DQM itself. The structure is platform-dependent, but typically includes listeners and trigger monitors, together with operator commands and panels.

A *message channel* is a one-way pipe for moving messages from one queue manager to another. Thus a message channel has two end-points, represented by a pair of MCAs. Each end point has a definition of its end of the message channel. For example, one end would define a sender, the other end a receiver.

For details of how to define channels, see:

- Windows Linux UNIX “Monitoring and controlling channels on Windows, UNIX and Linux platforms” on page 173
- z/OS “Monitoring and controlling channels on z/OS” on page 591
- IBM i “Monitoring and controlling channels on IBM i” on page 195

For message channel planning examples, see:

- Windows Linux UNIX Message channel planning example for distributed platforms
- IBM i Message channel planning example for IBM MQ for IBM i
- z/OS Message channel planning example for z/OS

- z/OS Message channel planning example for z/OS using queue-sharing groups

For information about channel exits, see Channel-exit programs for messaging channels.

Related concepts

"Message sending and receiving" on page 145
The following figure shows the distributed queue management model, detailing the relationships between entities when messages are transmitted. It also shows the flow for control.

"Channel control function" on page 152
The channel control function provides facilities for you to define, monitor, and control channels.

"What happens when a message cannot be delivered?" on page 165
When a message cannot be delivered, the MCA can process it in several ways. It can try again, it can return-to-sender, or it can put it on the dead-letter queue.

"Initialization and configuration files" on page 170
The handling of channel initialization data depends on your IBM MQ platform.

"Data conversion for messages" on page 171
IBM MQ messages might require data conversion when sent between queues on different queue managers.

"Writing your own message channel agents" on page 171
IBM MQ allows you to write your own message channel agent (MCA) programs or to install one from an independent software vendor.

"Other things to consider for distributed queue management" on page 172
Other topics to consider when preparing IBM MQ for distributed queue management. This topic covers Undelivered-message queue, Queues in use, System extensions and user-exit programs, and Running channels and listeners as trusted applications.

Related reference

Example configuration information

Message sending and receiving

The following figure shows the distributed queue management model, detailing the relationships between entities when messages are transmitted. It also shows the flow for control.

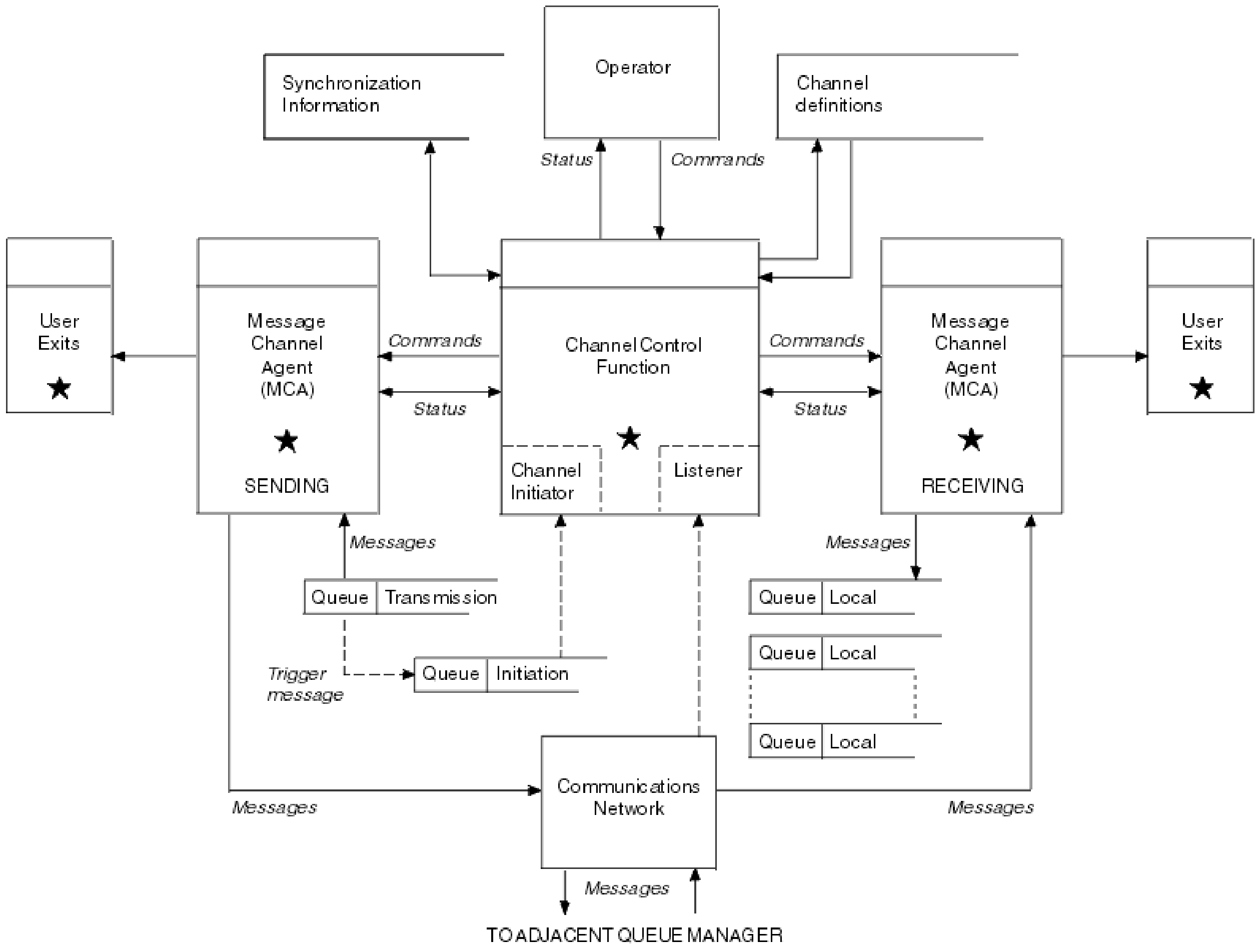

Figure 15. Distributed queue management model

Note:

1. There is one MCA per channel, depending on the platform. There might be one or more channel control functions for a particular queue manager.
2. The implementation of MCAs and channel control functions is highly platform-dependent. They can be programs or processes or threads, and they can be a single entity or many comprising several independent or linked parts.
3. All components marked with a star can use the MQI.

Channel parameters

An MCA receives its parameters in one of several ways:

- If started by a command, the channel name is passed in a data area. The MCA then reads the channel definition directly to obtain its attributes.
- For sender, and in some cases server channels, the MCA can be started automatically by the queue manager trigger. The channel name is retrieved from the trigger process definition, where applicable, and is passed to the MCA. The remaining processing is the same as previously described. Server channels must only be set up to trigger if they are fully qualified, that is, they specify a CONNAME to connect to.

- If started remotely by a sender, server, requester, or client-connection, the channel name is passed in the initial data from the partner message channel agent. The MCA reads the channel definition directly to obtain its attributes.

Certain attributes not defined in the channel definition are also negotiable:

Split messages
If one end does not support split messages then the split messages are not sent.

Conversion capability
If one end cannot perform the necessary code page conversion or numeric encoding conversion when needed, the other end must handle it. If neither end supports it, when needed, the channel cannot start.

Distribution list support
If one end does not support distribution lists, the partner MCA sets a flag in its transmission queue so that it knows to intercept messages intended for multiple destinations.

Channel status and sequence numbers

Message channel agent programs keep records of the current sequence number and logical unit of work number for each channel, and of the general status of the channel. Some platforms allow you to display this status information to help you control channels.

How to send a message to another queue manager

This section describes the simplest way to send a message between queue managers, including prerequisites and authorizations required. Other methods can also be used to send messages to a remote queue manager.

Before you send a message from one queue manager to another, you need to do the following steps:

1. Check that your chosen communication protocol is available.
2. Start the queue managers.
3. Start the channel initiators.
4. Start the listeners.

You also need to have the correct IBM MQ security authorization to create the objects required.

To send messages from one queue manager to another:

- Define the following objects on the source queue manager:
  - Sender channel
  - Remote queue definition
  - Initiation queue (z/OS required on z/OS, otherwise optional)
  - Transmission queue
  - Dead-letter queue
- Define the following objects on the target queue manager:
  - Receiver channel
  - Target queue
  - Dead-letter queue

You can use several different methods to define these objects, depending on your IBM MQ platform:

- On all platforms, you can use the IBM MQ script commands (MQSC) described in The MQSC commands the programmable command format (PCF) commands described in Automating administration tasks, or the IBM MQ Explorer.

- z/OS On z/OS, you can also use the Operation and Control panels described in Administering IBM MQ for z/OS .
- IBM i On IBM i, you can also use the panel interface.

See the following subtopics for more information on creating the components for sending messages to another queue manager:

Related concepts

"Creating and managing queue managers on distributed platforms" on page 5
Before you can use messages and queues, you must create and start at least one queue manager and its associated objects.

"IBM MQ distributed queuing techniques" on page 123
The subtopics in this section describe techniques that are of use when planning channels. These subtopics describe techniques to help you plan how to connect your queue managers together, and manage the flow of messages between your applications.

"Introduction to distributed queue management" on page 143
Distributed queue management (DQM) is used to define and control communication between queue managers.

"Triggering channels" on page 166
IBM MQ provides a facility for starting an application automatically when certain conditions on a queue are met. This facility is called triggering.

"Safety of messages" on page 164
In addition to the typical recovery features of IBM MQ, distributed queue management ensures that messages are delivered properly by using a sync point procedure coordinated between the two ends of the message channel. If this procedure detects an error, it closes the channel so that you can investigate the problem, and keeps the messages safely in the transmission queue until the channel is restarted.

"Monitoring and controlling channels on Windows, UNIX and Linux platforms" on page 173
For DQM you need to create, monitor, and control the channels to remote queue managers. You can control channels using commands, programs, IBM MQ Explorer, files for the channel definitions, and a storage area for synchronization information.

"Monitoring and controlling channels on IBM i" on page 195
Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers. Each queue manager has a DQM program for controlling interconnections to compatible remote queue managers.

"Configuring connections between the client and server" on page 14
To configure the communication links between IBM MQ MQI clients and servers, decide on your communication protocol, define the connections at both ends of the link, start a listener, and define channels.

"Setting up communications with other queue managers" on page 587
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

Related tasks

"Configuring a queue manager cluster" on page 215
Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. You can define cluster components, and create and manage clusters.

Defining the channels

To send messages from one queue manager to another, you must define two channels. You must define one channel on the source queue manager and one channel on the target queue manager.

On the source queue manager

Define a channel with a channel type of SENDER. You need to specify the following:

- The name of the transmission queue to be used (the XMITQ attribute).
- The connection name of the partner system (the CONNAME attribute).
- The name of the communication protocol you are using (the TRPTYPE attribute). On IBM MQ for z/OS, the protocol must be TCP or LU6.2. On other platforms, you do not have to specify this. You can leave it to pick up the value from your default channel definition.

Details of all the channel attributes are given in Channel attributes.

On the target queue manager
Define a channel with a channel type of RECEIVER, and the same name as the sender channel.

Specify the name of the communication protocol you are using (the TRPTYPE attribute). On IBM MQ for z/OS, the protocol must be TCP or LU6.2. On other platforms, you do not have to specify this. You can leave it to pick up the value from your default channel definition.

Receiver channel definitions can be generic. This means that if you have several queue managers communicating with the same receiver, the sending channels can all specify the same name for the receiver, and one receiver definition applies to them all.

When you have defined the channel, you can test it using the PING CHANNEL command. This command sends a special message from the sender channel to the receiver channel and checks that it is returned.

Note: The value of the TRPTYPE parameter is ignored by the responding message channel agent. For example, a TRPTYPE of TCP on the sender channel definition successfully starts with a TRPTYPE of LU62 on the receiver channel definition as a partner.

Defining the queues

To send messages from one queue manager to another, you must define up to six queues. You must define up to four queues on the source queue manager, and up to two queues on the target queue manager.

On the source queue manager

- Remote queue definition

  In this definition, specify the following:

  Remote queue manager name
  The name of the target queue manager.

  Remote queue name
  The name of the target queue on the target queue manager.

  Transmission queue name
  The name of the transmission queue. You do not have to specify this transmission queue name. If you do not, a transmission queue with the same name as the target queue manager is used. If this does not exist, the default transmission queue is used. You are advised to give the transmission queue the same name as the target queue manager so that the queue is found by default.
- Initiation queue definition

  z/OS This is required. You must use the initiation queue called SYSTEM.CHANNEL.INITQ.

  IBM i distributed This is optional. Consider naming the initiation queue SYSTEM.CHANNEL.INITQ.
- Transmission queue definition

  A local queue with the USAGE attribute set to XMITQ. IBM i If you are using the IBM MQ for IBM i native interface, the USAGE attribute is *TMQ.
- Dead-letter queue definition

  Define a dead-letter queue to which undelivered messages can be written.

On the target queue manager

- Local queue definition

  The target queue. The name of this queue must be the same as that specified in the remote queue name field of the remote queue definition on the source queue manager.
- Dead-letter queue definition

  Define a dead-letter queue to which undelivered messages can be written.

Related concepts

"Creating a transmission queue" on page 149
Before a channel (other than a requester channel) can be started, the transmission queue must be defined as described in this section. The transmission queue must be named in the channel definition.

"Creating a transmission queue on IBM i" on page 149
You can create a transmission queue on the IBM i platform by using the Create MQM Queue panel.

Creating a transmission queue

Before a channel (other than a requester channel) can be started, the transmission queue must be defined as described in this section. The transmission queue must be named in the channel definition.

Define a local queue with the USAGE attribute set to XMITQ for each sending message channel. If you want to use a specific transmission queue in your remote queue definitions, create a remote queue as shown.

To create a transmission queue, use the IBM MQ Commands (MQSC), as shown in the following examples:

Create transmission queue example

```
DEFINE QLOCAL(QM2) DESCR('Transmission queue to QM2') USAGE(XMITQ)
```

Create remote queue example

```
DEFINE QREMOTE(PAYROLL) DESCR('Remote queue for QM2') +
XMITQ(QM2) RNAME(PAYROLL) RQMNAME(QM2)
```

Consider naming the transmission queue the queue manager name on the remote system, as shown in the examples.

Creating a transmission queue on IBM i

You can create a transmission queue on the IBM i platform by using the Create MQM Queue panel.

You must define a local queue with the Usage field attribute set to *TMQ, for each sending message channel.

If you want to use remote queue definitions, use the same command to create a queue of type *RMT, and Usage of *NORMAL.

To create a transmission queue, use the CRTMQMQ command from the command line to present you with the first queue creation panel; see Figure 16 on page 150.

```
Create MQM Queue (CRTMQMQ)

Type choices, press Enter.

Queue name . . . . . . . . . . .

Queue type . . . . . . . . . . .  ____     *ALS, *LCL, *MDL, *RMT

Message Queue Manager name . . .  *DFT________________________________
-----








Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
+
```

Figure 16. Create a queue (1)

Type the name of the queue and specify the type of queue that you want to create: Local, Remote, or Alias. For a transmission queue, specify Local (*LCL) on this panel and press enter.

You are presented with the second page of the Create MQM Queue panel; see Figure 17 on page 150.

```
Create MQM Queue (CRTMQMQ)

Type choices, press Enter.

Queue name . . . . . . . . . . . > HURS.2.HURS.PRIORIT

Queue type . . . . . . . . . . . > *LCL      *ALS, *LCL, *MDL, *RMT
Message Queue Manager name . . .  *DFT
Replace . . . . . . . . . . . .  *NO      *NO, *YES
Text 'description' . . . . . . .  '                  '
Put enabled . . . . . . . . . .  *YES     *SYSDFTQ, *NO, *YES
Default message priority . . . .  0       0-9, *SYSDFTQ
Default message persistence . .  *NO      *SYSDFTQ, *NO, *YES
Process name . . . . . . . . . .  '                  '
Triggering enabled . . . . . . .  *NO      *SYSDFTQ, *NO, *YES
Get enabled . . . . . . . . . .  *YES     *SYSDFTQ, *NO, *YES
Sharing enabled . . . . . . . .  *YES     *SYSDFTQ, *NO, *YES




More...
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 17. Create a queue (2)

Change any of the default values shown. Press page down to scroll to the next screen; see Figure 18 on page 151.

```
Create MQM Queue (CRTMQMQ)

Type choices, press Enter.

Default share option . . . . . .  *YES     *SYSDFTQ, *NO, *YES
Message delivery sequence . . .  *PTY     *SYSDFTQ, *PTY, *FIFO
Harden backout count . . . . . .  *NO      *SYSDFTQ, *NO, *YES
Trigger type . . . . . . . . . .  *FIRST    *SYSDFTQ, *FIRST, *ALL...
Trigger depth . . . . . . . . .  1       1-999999999, *SYSDFTQ
Trigger message priority . . . .  0       0-9, *SYSDFTQ
Trigger data . . . . . . . . . .  '                 '
Retention interval . . . . . . .  999999999   0-999999999, *SYSDFTQ
Maximum queue depth . . . . . .  5000     1-24000, *SYSDFTQ
Maximum message length . . . . .  4194304    0-4194304, *SYSDFTQ
Backout threshold . . . . . . .  0       0-999999999, *SYSDFTQ
Backout requeue queue . . . . .  '                 '
Initiation queue . . . . . . . .  '                 '



More...
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 18. Create a queue (3)

Type *TMQ, for transmission queue, in the Usage field of this panel, and change any of the default values shown in the other fields.

```
Create MQM Queue (CRTMQMQ)

Type choices, press Enter.

Usage . . . . . . . . . . . . .  *TMQ     *SYSDFTQ, *NORMAL, *TMQ
Queue depth high threshold . . .  80      0-100, *SYSDFTQ
Queue depth low threshold . . .  20      0-100, *SYSDFTQ
Queue full events enabled . . .  *YES     *SYSDFTQ, *NO, *YES
Queue high events enabled . . .  *YES     *SYSDFTQ, *NO, *YES
Queue low events enabled . . . .  *YES     *SYSDFTQ, *NO, *YES
Service interval . . . . . . . .  999999999   0-999999999, *SYSDFTQ
Service interval events . . . .  *NONE     *SYSDFTQ, *HIGH, *OK, *NONE
Distribution list support . . .  *NO      *SYSDFTQ, *NO, *YES
Cluster Name . . . . . . . . . .  *SYSDFTQ
Cluster Name List . . . . . . .  *SYSDFTQ
Default Binding . . . . . . . .  *SYSDFTQ   *SYSDFTQ, *OPEN, *NOTFIXED





Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 19. Create a queue (4)

When you are satisfied that the fields contain the correct data, press enter to create the queue.

Starting the channel

When you put messages on the remote queue defined at the source queue manager, they are stored on the transmission queue until the channel is started. When the channel has been started, the messages are delivered to the target queue on the remote queue manager.

Start the channel on the sending queue manager using the START CHANNEL command. When you start the sending channel, the receiving channel is started automatically (by the listener) and the messages are sent to the target queue. Both ends of the message channel must be running for messages to be transferred.

Because the two ends of the channel are on different queue managers, they could have been defined with different attributes. To resolve any differences, there is an initial data negotiation between the two ends when the channel starts. In general, the two ends of the channel operate with the attributes needing the fewer resources. This enables larger systems to accommodate the lesser resources of smaller systems at the other end of the message channel.

The sending MCA splits large messages before sending them across the channel. They are reassembled at the remote queue manager. This is not apparent to the user.

An MCA can transfer messages using multiple threads. This process, called *pipelining* enables the MCA to transfer messages more efficiently, with fewer wait states. Pipelining improves channel performance.

Channel control function

The channel control function provides facilities for you to define, monitor, and control channels.

Commands are issued through panels, programs, or from a command line to the channel control function. The panel interface also displays channel status and channel definition data. You can use Programmable Command Formats or those IBM MQ commands (MQSC) and control commands that are detailed in "Monitoring and controlling channels on Windows, UNIX and Linux platforms" on page 173.

The commands fall into the following groups:

- Channel administration
- Channel control
- Channel status monitoring

Channel administration commands deal with the definitions of the channels. They enable you to:

- Create a channel definition
- Copy a channel definition
- Alter a channel definition
- Delete a channel definition

Channel control commands manage the operation of the channels. They enable you to:

- Start a channel
- Stop a channel
- Re-synchronize with partner (in some implementations)
- Reset message sequence numbers
- Resolve an in-doubt batch of messages
- Ping; send a test communication across the channel

Channel monitoring displays the state of channels, for example:

- Current channel settings
- Whether the channel is active or inactive
- Whether the channel terminated in a synchronized state

For more information about defining, controlling and monitoring channels, see the following subtopics:

Preparing channels

Before trying to start a message channel or MQI channel, you must prepare the channel. You must make sure that all the attributes of the local and remote channel definitions are correct and compatible.

Channel attributes describes the channel definitions and attributes.

Although you set up explicit channel definitions, the channel negotiations carried out when a channel starts, might override one or other of the values defined. This behavior is normal, and not apparent to the user, and has been arranged in this way so that otherwise incompatible definitions can work together.

Auto-definition of receiver and server-connection channels

In IBM MQ on all platforms except z/OS, if there is no appropriate channel definition, then for a receiver or server-connection channel that has auto-definition enabled, a definition is created automatically. The definition is created using:

1. The appropriate model channel definition, SYSTEM.AUTO.RECEIVER, or SYSTEM.AUTO.SVRCONN. The model channel definitions for auto-definition are the same as the system defaults, SYSTEM.DEF.RECEIVER, and SYSTEM.DEF.SVRCONN, except for the description field, which is "Auto-defined by" followed by 49 blanks. The systems administrator can choose to change any part of the supplied model channel definitions.
2. Information from the partner system. The values from the partner are used for the channel name and the sequence number wrap value.
3. A channel exit program, which you can use to alter the values created by the auto-definition. See Channel auto-definition exit program.

The description is then checked to determine whether it has been altered by an auto-definition exit or because the model definition has been changed. If the first 44 characters are still "Auto-defined by" followed by 29 blanks, the queue manager name is added. If the final 20 characters are still all blanks the local time and date are added.

When the definition has been created and stored the channel start proceeds as though the definition had always existed. The batch size, transmission size, and message size are negotiated with the partner.

Defining other objects

Before a message channel can be started, both ends must be defined (or enabled for auto-definition) at their queue managers. The transmission queue it is to serve must be defined to the queue manager at the sending end. The communication link must be defined and available. It might be necessary for you to prepare other IBM MQ objects, such as remote queue definitions, queue manager alias definitions, and reply-to queue alias definitions, to implement the scenarios described in “Configuring distributed queuing” on page 122.

For information about defining MQI channels, see “Defining MQI channels” on page 28.

Multiple message channels per transmission queue

It is possible to define more than one channel per transmission queue, but only one of these channels can be active at any one time. Consider this option for the provision of alternative routes between queue managers for traffic balancing and link failure corrective action. A transmission queue cannot be used by another channel if the previous channel to use it terminated leaving a batch of messages in-doubt at the sending end. For more information, see “In-doubt channels” on page 163.

Starting a channel

A channel can be caused to start transmitting messages in one of four ways. It can be:

- Started by an operator (not receiver, cluster-receiver, or server-connection channels).
- Triggered from the transmission queue. This method applies to sender channels and fully qualified server channels (those channels which specify a CONNAME) only. You must prepare the necessary objects for triggering channels.
- Started from an application program (not receiver, cluster-receiver, or server-connection channels).
- Started remotely from the network by a sender, cluster-sender, requester, server, or client-connection channel. Receiver, cluster-receiver, and possibly server and requester channel transmissions, are

started this way; so are server-connection channels. The channels themselves must already be started (that is, enabled).

Note: Because a channel is 'started' it is not necessarily transmitting messages. Instead, it might be 'enabled' to start transmitting when one of the four events previously described occurs. The enabling and disabling of a channel is achieved using the START and STOP operator commands.

Channel states

A channel can be in one of many states at any time. Some states also have substates. From a given state a channel can move into other states.

Figure 20 on page 154 shows the hierarchy of all possible channel states and the substates that apply to each of the channel states.

Figure 21 on page 155 shows the links between channel states. These links apply to all types of message channel and server-connection channels.

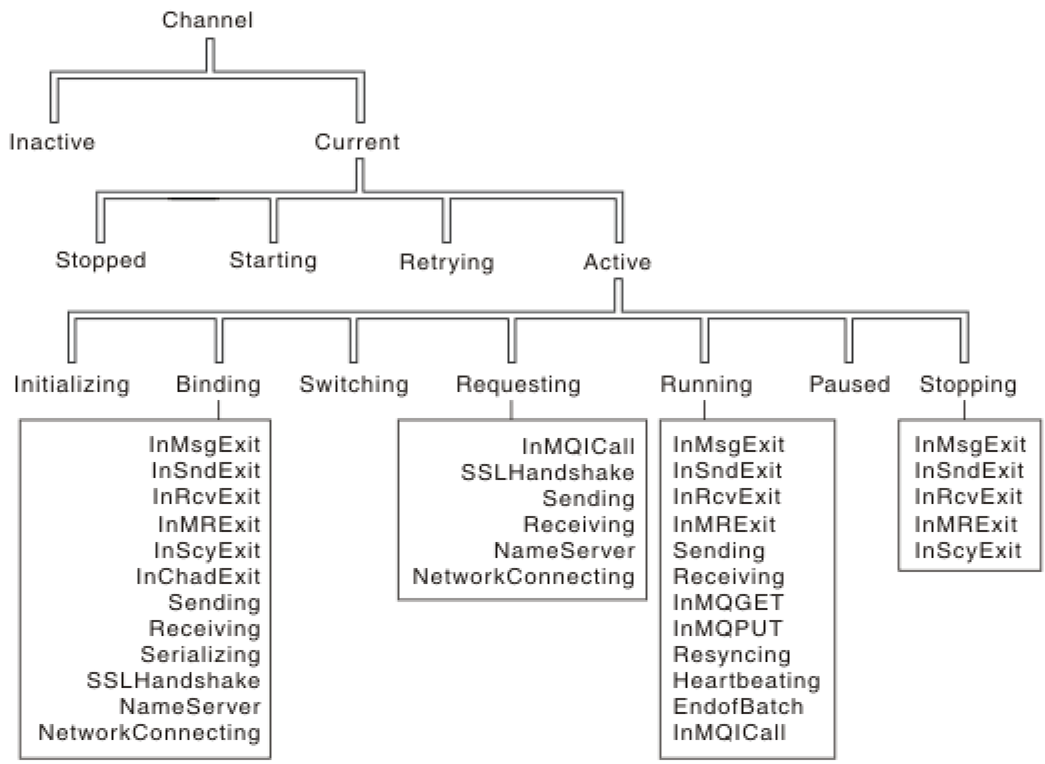

Figure 20. Channel states and substates

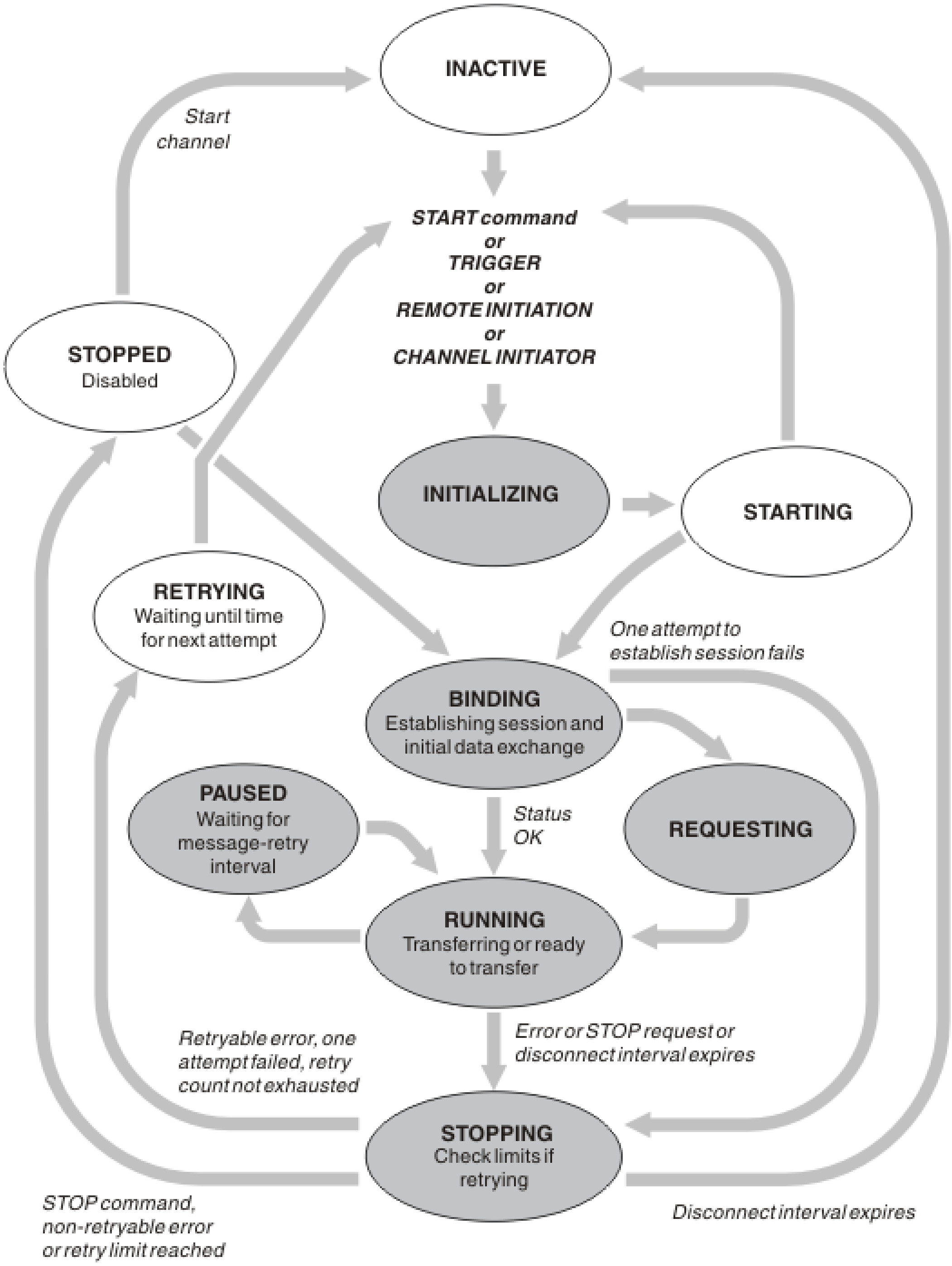

Figure 21. Flows between channel states

Current and active

A channel is *current* if it is in any state other than inactive. A current channel is *active* unless it is in RETRYING, STOPPED, or STARTING state. When a channel is active, it is consuming resource and a process or thread is running. The seven possible states of an active channel (INITIALIZING, BINDING, SWITCHING, REQUESTING, RUNNING, PAUSED, or STOPPING) are highlighted in Figure 21 on page 155.

An active channel can also show a substate giving more detail of exactly what the channel is doing. The substates for each state are shown in Figure 20 on page 154.

Current and active
The channel is "current" if it is in any state other than inactive. A current channel is "active" unless it is in RETRYING, STOPPED, or STARTING state.

If a channel is "active" it might also show a substate giving more detail of exactly what the channel is doing.

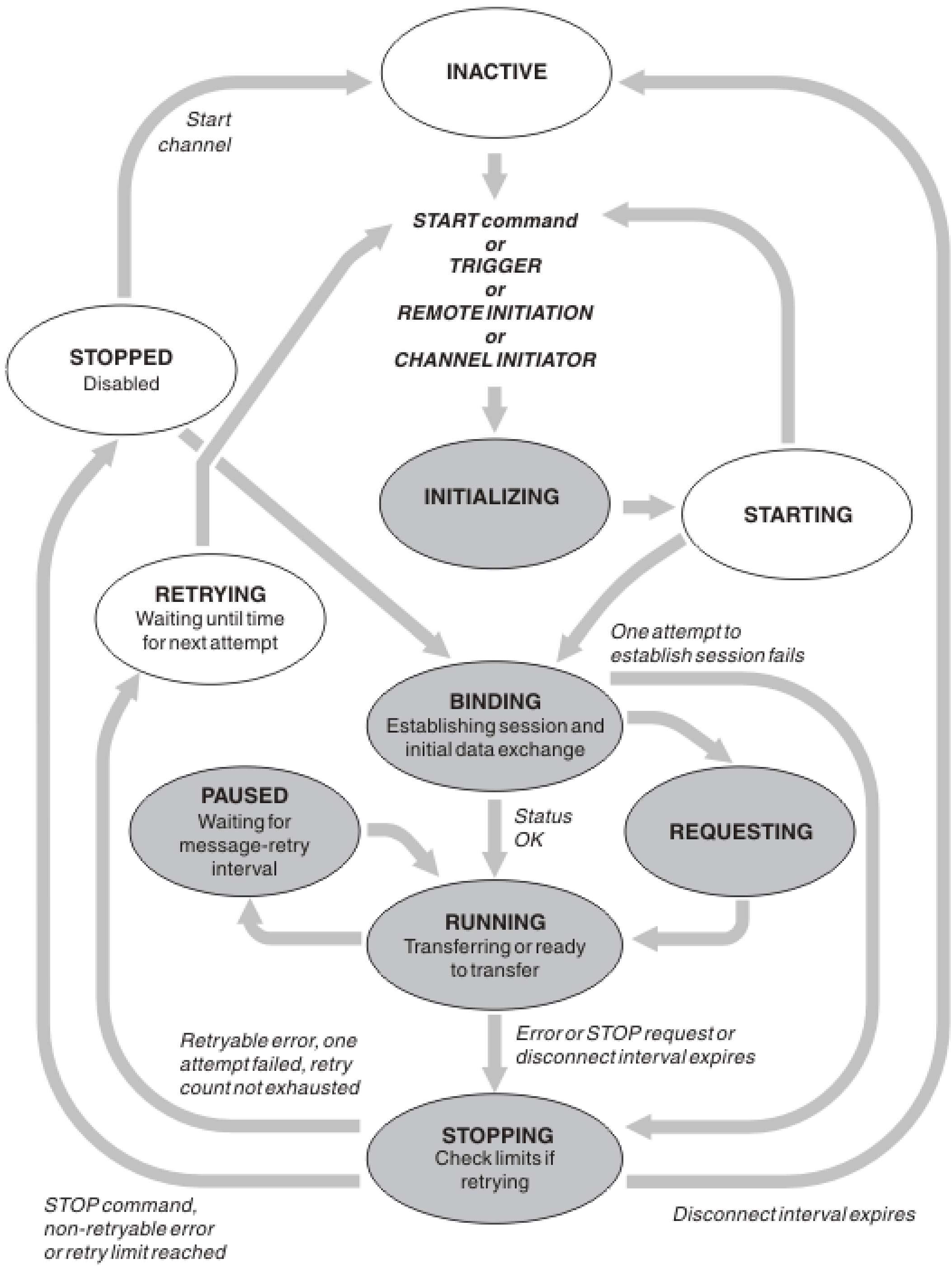

Figure 22. Flows between channel states

Note:

1. When a channel is in one of the six states highlighted in Figure 22 on page 156 (INITIALIZING, BINDING, REQUESTING, RUNNING, PAUSED, or STOPPING), it is consuming resource and a process or thread is running; the channel is *active*.

2. When a channel is in STOPPED state, the session might be active because the next state is not yet known.

Specifying the maximum number of current channels

You can specify the maximum number of channels that can be current at one time. This number is the number of channels that have entries in the channel status table, including channels that are retrying and channels that are stopped. Specify this using z/OS `ALTER QMGR MAXCHL` for z/OS, IBM i the queue manager initialization file for IBM i, the queue manager configuration file for UNIX and Linux systems, or the MQ Explorer. For more information about the values set using the initialization or the configuration file see Configuration file stanzas for distributed queuing. For more information about specifying the maximum number of channels, see Administering IBM MQ for IBM MQ for UNIX and Linux systems, and Windows systems IBM i, Administering IBM i for IBM MQ for IBM i z/OS, or Administering IBM MQ for z/OS for IBM MQ for z/OS .

Note:

1. Server-connection channels are included in this number.
2. A channel must be current before it can become active. If a channel is started, but cannot become current, the start fails.

Specifying the maximum number of active channels

You can also specify the maximum number of active channels to prevent your system being overloaded by many starting channels. If you use this method, set the disconnect interval attribute to a low value to allow waiting channels to start as soon as other channels terminate.

Each time a channel that is retrying attempts to establish connection with its partner, it must become an active channel. If the attempt fails, it remains a current channel that is not active, until it is time for the next attempt. The number of times that a channel retries, and how often, is determined by the retry count and retry interval channel attributes. There are short and long values for both these attributes. See Channel attributes for more information.

When a channel has to become an active channel (because a START command has been issued, or because it has been triggered, or because it is time for another retry attempt), but is unable to do so because the number of active channels is already at the maximum value, the channel waits until one of the active slots is freed by another channel instance ceasing to be active. If, however, a channel is starting because it is being initiated remotely, and there are no active slots available for it at that time, the remote initiation is rejected.

Whenever a channel, other than a requester channel, is attempting to become active, it goes into the STARTING state. This state occurs even if there is an active slot immediately available, although it is only in the STARTING state for a short time. However, if the channel has to wait for an active slot, it is in STARTING state while it is waiting.

Requester channels do not go into STARTING state. If a requester channel cannot start because the number of active channels is already at the limit, the channel ends abnormally.

Whenever a channel, other than a requester channel, is unable to get an active slot, and so waits for one, a message is written to the log z/OS or the z/OS console, and an event is generated. When a slot is later freed and the channel is able to acquire it, another message and event are generated. Neither of these events and messages are generated if the channel is able to acquire a slot straight away.

If a STOP CHANNEL command is issued while the channel is waiting to become active, the channel goes to STOPPED state. A Channel-Stopped event is raised.

Server-connection channels are included in the maximum number of active channels.

For more information about specifying the maximum number of active channels, see Administering IBM MQ for IBM MQ for UNIX and Linux systems, and Windows systems IBM i, Administering IBM i for IBM MQ for IBM i z/OS, or Administering IBM MQ for z/OS for IBM MQ for z/OS .

Channel errors
Errors on channels cause the channel to stop further transmissions. If the channel is a sender or server, it goes to RETRY state because it is possible that the problem might clear itself. If it cannot go to RETRY state, the channel goes to STOPPED state.

For sending channels, the associated transmission queue is set to GET(DISABLED) and triggering is turned off. (A STOP command with STATUS(STOPPED) takes the side that issued it to STOPPED state; only expiry of the disconnect interval or a STOP command with STATUS(INACTIVE) makes it end normally and become inactive.) Channels that are in STOPPED state need operator intervention before they can restart (see "Restarting stopped channels" on page 162).

Note: For IBM i IBM i, UNIX, Linux and Windows systems, a channel initiator must be running for retry to be attempted. If the channel initiator is not available, the channel becomes inactive and must be manually restarted. If you are using a script to start the channel, ensure that the channel initiator is running before you try to run the script.

Long retry count (LONGRTY) describes how retrying works. If the error clears, the channel restarts automatically, and the transmission queue is re-enabled. If the retry limit is reached without the error clearing, the channel goes to STOPPED state. A stopped channel must be restarted manually by the operator. If the error is still present, it does not retry again. When it does start successfully, the transmission queue is re-enabled.

z/OS If the channel initiator stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the channel initiator is restarted. However, the channel status for the SVRCONN channel type is reset if the channel initiator stops while the channel is in STOPPED status.

IBM i distributed If the queue manager stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the queue manager is restarted. From IBM MQ 8.0 onwards, this applies to SVRCONN channels as well. Previously, the channel status for the SVRCONN channel type was reset if the channel initiator stopped while the channel was in STOPPED status.

If a channel is unable to put a message to the target queue because that queue is full or put inhibited, the channel can retry the operation a number of times (specified in the message-retry count attribute) at a time interval (specified in the message-retry interval attribute). Alternatively, you can write your own message-retry exit that determines which circumstances cause a retry, and the number of attempts made. The channel goes to PAUSED state while waiting for the message-retry interval to finish.

See Channel attributes for information about the channel attributes, and Channel-exit programs for messaging channels for information about the message-retry exit.

Server-connection channel limits

You can set server-connection channel limits to prevent client applications from exhausting queue manager channel resources, **MAXINST**, and to prevent a single client application from exhausting server-connection channel capacity, **MAXINSTC**.

You set **MAXINST** and **MAXINSTC** with the **DEFINE CHANNEL** command.

A maximum total number of channels can be active at any time on an individual queue manager. The total number of server-connection channel instances are included in the maximum number of active channels.

If you do not specify the maximum number of simultaneous instances of a server-connection channel that can be started, then it is possible for a single client application, connecting to a single server-connection channel, to exhaust the maximum number of active channels that are available. When the maximum number of active channels is reached, it prevents any other channels from being started on the queue manager. To avoid this situation, you must limit the number of simultaneous instances of an individual server-connection channel that can be started, regardless of which client started them.

If the value of the limit is reduced to below the currently running number of instances of the server connection channel, even to zero, then the running channels are not affected. New instances cannot be started until sufficient existing instances have ceased to run so that the number of currently running instances is less than the value of the limit.

Also, many different client-connection channels can connect to an individual server-connection channel. The limit on the number of simultaneous instances of an individual server-connection channel that can be started, regardless of which client started them, prevents any client from exhausting the maximum active channel capacity of the queue manager. If you do not also limit the number of simultaneous instances of an individual server-connection channel that can be started from an individual client, then it is possible for a single, faulty client application to open so many connections that it exhausts the channel capacity allocated to an individual server-connection channel, and therefore prevents other clients that need to use the channel from connecting to it. To avoid this situation, you must limit the number of simultaneous instances of an individual server-connection channel that can be started from an individual client.

If the value of the individual client limit is reduced below the number of instances of the server-connection channel that are currently running from individual clients, even to zero, then the running channels are not affected. However, new instances of the server-connection channel cannot be started from an individual client that exceeds the new limit until sufficient existing instances from that client have ceased to run so that the number of currently running instances is less than the value of this parameter.

Related reference

Channel attributes and channel types

DEFINE CHANNEL

Checking that the other end of the channel is still available

You can use the heartbeat interval, the keep alive interval, and the receive timeout, to check that the other end of the channel is available.

Heartbeats

You can use the heartbeat interval channel attribute to specify that flows are to be passed from the sending MCA when there are no messages on the transmission queue, as is described in Heartbeat interval (HBINT).

Keep alive

In IBM MQ for z/OS, if you are using TCP/IP as the transport protocol, you can also specify a value for the **Keepalive** interval channel attribute (KAINT). You are recommended to give the **Keepalive** interval a higher value than the heartbeat interval, and a smaller value than the disconnect value. You can use this attribute to specify a time-out value for each channel, as is described in Keepalive Interval (KAINT).

In IBM MQ for IBM i, UNIX, Linux, and Windows systems, if you are using TCP as your transport protocol, you can set `keepalive=yes`. If you specify this option, TCP periodically checks that the other end of the connection is still available. It is not, the channel is terminated. This option is described in Keepalive Interval (KAINT).

If you have unreliable channels that report TCP errors, use of the **Keepalive** option means that your channels are more likely to recover.

You can specify time intervals to control the behavior of the **Keepalive** option. When you change the time interval, only TCP/IP channels started after the change are affected. Ensure that the value that you choose for the time interval is less than the value of the disconnect interval for the channel.

For more information about using the **Keepalive** option, see the **KAINT** parameter in the **DEFINE CHANNEL** command.

Receive timeout

If you are using TCP as your transport protocol, the receiving end of an idle non-MQI channel connection is also closed if no data is received for a period. This period, the *receive time-out* value, is determined according to the HBINT (heartbeat interval) value.

In IBM MQ for IBM i, UNIX, Linux, and Windows systems, the *receive time-out* value is set as follows:

1. For an initial number of flows, before any negotiation takes place, the *receive time-out* value is twice the HBINT value from the channel definition.
2. After the channels negotiate an HBINT value, if HBINT is set to less than 60 seconds, the *receive time-out* value is set to twice this value. If HBINT is set to 60 seconds or more, the *receive time-out* value is set to 60 seconds greater than the value of HBINT.

In IBM MQ for z/OS, the *receive time-out* value is set as follows:

1. For an initial number of flows, before any negotiation takes place, the *receive time-out* value is twice the HBINT value from the channel definition.
2. If RCVTIME is set, the timeout is set to one of

   - the negotiated HBINT multiplied by a constant
   - the negotiated HBINT plus a constant number of seconds
   - a constant number of seconds

   depending on the RCVTTYPE parameter, and subject to any limit imposed by RCVTMIN if it applies. RCVTMIN does not apply when RCVTTYPE(EQUAL) is configured. If you use a constant value of RCVTIME and you use a heartbeat interval, do not specify an RCVTIME less than the heartbeat interval. For details of the RCVTIME, RCVTMIN and RCVTTYPE attributes, see the ALTER QMGR command.

Note:

1. If either of the values is zero, there is no timeout.
2. For connections that do not support heartbeats, the HBINT value is negotiated to zero in step 2 and hence there is no timeout, so you must use TCP/IP KEEPALIVE.
3. For client connections that use sharing conversations, heartbeats can flow across the channel (from both ends) all the time, not just when an MQGET is outstanding.
4. For client connections where sharing conversations are not in use, heartbeats are flowed from the server only when the client issues an MQGET call with wait. Therefore, you are not recommended to set the heartbeat interval too small for client channels. For example, if the heartbeat is set to 10 seconds, an MQCMIT call fails (with MQRC_CONNECTION_BROKEN) if it takes longer than 20 seconds to commit because no data flowed during this time. This can happen with large units of work. However, it does not happen if appropriate values are chosen for the heartbeat interval because only MQGET with wait takes significant periods of time.

   Provided SHARECNV is not zero, the client uses a full duplex connection, which means that the client can (and does) heartbeat during all MQI calls
5. In IBM MQ Version 7 Client channels, heartbeats can flow from both the server as well as the client side. The timeout at either end is based upon 2*HBINT for HBINTs of less than 60 seconds and HBINT+60 for HBINTs of over 60 seconds.
6. Canceling the connection after twice the heartbeat interval is valid because a data or heartbeat flow is expected at least at every heartbeat interval. Setting the heartbeat interval too small, however, can cause problems, especially if you are using channel exits. For example, if the HBINT value is one second, and a send or receive exit is used, the receiving end waits for only 2 seconds before canceling the channel. If the MCA is performing a task such as encrypting the message, this value might be too short.

Adopting an MCA

The Adopt MCA function enables IBM MQ to cancel a receiver channel and start a new one in its place.

If a channel loses contact, the receiver channel can be left in a 'communications receive' state. When communications are re-established the sender channel attempts to reconnect. If the remote queue manager finds that the receiver channel is already running it does not allow another version of the same receiver channel to start. This problem requires user intervention to rectify the problem or the use of system keepalive.

The Adopt MCA function solves the problem automatically. It enables IBM MQ to cancel a receiver channel and to start a new one in its place.

The function can be set up with various options:

- distributed For distributed platforms, see Administering IBM MQ.
- z/OS For z/OS, see Administering IBM MQ for z/OS.
- IBM i For IBM i, see Administering IBM i .

Stopping and quiescing channels

You can stop and quiesce a channel before the disconnect time interval expires.

Message channels are designed to be long-running connections between queue managers with orderly termination controlled only by the disconnect interval channel attribute. This mechanism works well unless the operator needs to terminate the channel before the disconnect time interval expires. This need can occur in the following situations:

- System quiesce
- Resource conservation
- Unilateral action at one end of a channel

In this case, you can stop the channel. You can do this using:

- the STOP CHANNEL MQSC command
- the Stop Channel PCF command
- the IBM MQ Explorer
- IBM i z/OS other platform-specific mechanisms, as follows:

  z/OS **For z/OS:**
  : The Stop a channel panel

  IBM i **For IBM i:**
  : The ENDMQMCHL CL command or the END option on the WRKMQMCHL panel

There are three options for stopping channels using these commands:

QUIESCE
: The QUIESCE option attempts to end the current batch of messages before stopping the channel.

FORCE
: The FORCE option attempts to stop the channel immediately and might require the channel to resynchronize when it restarts because the channel might be left in doubt.

  z/OS On IBM MQ for z/OS, FORCE interrupts any message reallocation in progress, which might leave BIND_NOT_FIXED messages partially reallocated or out of order.

TERMINATE
: The TERMINATE option attempts to stop the channel immediately, and terminates the thread or process of the channel.

z/OS On IBM MQ for z/OS, TERMINATE interrupts any message reallocation in progress, which might leave BIND_NOT_FIXED messages partially reallocated or out of order.

All these options leave the channel in a STOPPED state, requiring operator intervention to restart it.

Stopping the channel at the sending end is effective but does require operator intervention to restart. At the receiving end of the channel, things are much more difficult because the MCA is waiting for data from the sending side, and there is no way to initiate an *orderly* termination of the channel from the receiving side; the stop command is pending until the MCA returns from its wait for data.

Consequently there are three recommended ways of using channels, depending upon the operational characteristics required:

- If you want your channels to be long running, note that there can be orderly termination only from the sending end. When channels are interrupted, that is, stopped, operator intervention (a START CHANNEL command) is required in order to restart them.
- If you want your channels to be active only when there are messages for them to transmit, set the disconnect interval to a fairly low value. The default setting is high and so is not recommended for channels where this level of control is required. Because it is difficult to interrupt the receiving channel, the most economical option is to have the channel automatically disconnect and reconnect as the workload demands. For most channels, the appropriate setting of the disconnect interval can be established heuristically.
- You can use the heartbeat-interval attribute to cause the sending MCA to send a heartbeat flow to the receiving MCA during periods in which it has no messages to send. This action releases the receiving MCA from its wait state and gives it an opportunity to quiesce the channel without waiting for the disconnect interval to expire. Give the heartbeat interval a lower value than the value of the disconnect interval.

  Note:

  1. You are advised to set the disconnect interval to a low value, or to use heartbeats, for server channels. This low value is to allow for the case where the requester channel ends abnormally (for example, because the channel was canceled) when there are no messages for the server channel to send. If the disconnect interval is set high and heartbeats are not in use, the server does not detect that the requester has ended (which it will only do the next time it tries to send a message to the requester). While the server is still running, it holds the transmission queue open for exclusive input in order to get any more messages that arrive on the queue. If an attempt is made to restart the channel from the requester, the start request receives an error because the server still has the transmission queue open for exclusive input. It is necessary to stop the server channel, and then restart the channel from the requester again.

Restarting stopped channels

When a channel goes into STOPPED state, you have to restart the channel manually.

You can restart the channel in the following ways:

- By using the **`START CHANNEL`** MQSC command.
- By using the **`Start Channel`** PCF command.
- By using the MQ Explorer.
- z/OS On z/OS, by using the Start a channel panel.
- IBM i On IBM i, by using the **`STRMQMCHL CL`** command or the **START** option on the WRKMQMCHL panel.

For sender or server channels, when the channel entered the STOPPED state, the associated transmission queue was set to GET(DISABLED) and triggering was set off. When the start request is received, these attributes are reset automatically.

z/OS If the channel initiator stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the channel initiator is restarted. However, the channel status for the SVRCONN channel type is reset if the channel initiator stops while the channel is in STOPPED status.

IBM i distributed If the queue manager stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the queue manager is restarted. From IBM MQ Version 8.0 onwards, this applies to SVRCONN channels as well. Previously, the channel status for the SVRCONN channel type was reset if the channel initiator stopped while the channel was in STOPPED status.

In-doubt channels

An in-doubt channel is a channel that is in doubt with a remote channel about which messages have been sent and received.

Note the distinction between this and a queue manager being in doubt about which messages should be committed to a queue.

You can reduce the opportunity for a channel to be placed in doubt by using the Batch Heartbeat channel parameter (BATCHHB). When a value for this parameter is specified, a sender channel checks that the remote channel is still active before taking any further action. If no response is received the receiver channel is considered to be no longer active. The messages can be rolled-back, and re-routed, and the sender-channel is not put in doubt. This reduces the time when the channel could be placed in doubt to the period between the sender channel verifying that the receiver channel is still active, and verifying that the receiver channel has received the sent messages. See Channel attributes for more information about the batch heartbeat parameter.

In-doubt channel problems are typically resolved automatically. Even when communication is lost, and a channel is placed in doubt with a message batch at the sender with receipt status unknown, the situation is resolved when communication is re-established. Sequence number and LUWID records are kept for this purpose. The channel is in doubt until LUWID information has been exchanged, and only one batch of messages can be in doubt for the channel.

You can, when necessary, resynchronize the channel manually. The term *manual* includes use of operators or programs that contain IBM MQ system management commands. The manual resynchronization process works as follows. This description uses MQSC commands, but you can also use the PCF equivalents.

1. Use the DISPLAY CHSTATUS command to find the last-committed logical unit of work ID (LUWID) for ***each*** side of the channel. Do this using the following commands:
   - For the in-doubt side of the channel:

     ```
     DISPLAY CHSTATUS( name ) SAVED CURLUWID
     ```

     You can use the CONNAME and XMITQ parameters to further identify the channel.
   - For the receiving side of the channel:

     ```
     DISPLAY CHSTATUS( name ) SAVED LSTLUWID
     ```

     You can use the CONNAME parameter to further identify the channel.

   The commands are different because only the sending side of the channel can be in doubt. The receiving side is never in doubt.

   On IBM MQ for IBM i, the DISPLAY CHSTATUS command can be executed from a file using the STRMQMMQSC command or the Work with MQM Channel Status CL command, WRKMQMCHST
2. If the two LUWIDs are the same, the receiving side has committed the unit of work that the sender considers to be in doubt. The sending side can now remove the in-doubt messages from the transmission queue and re-enable it. This is done with the following channel RESOLVE command:

```
RESOLVE CHANNEL( name ) ACTION(COMMIT)
```

3. If the two LUWIDs are different, the receiving side has not committed the unit of work that the sender considers to be in doubt. The sending side needs to retain the in-doubt messages on the transmission queue and re-send them. This is done with the following channel RESOLVE command:

```
RESOLVE CHANNEL( name ) ACTION(BACKOUT)
```

IBM i On IBM MQ for IBM i, you can use the Resolve MQM Channel command, RSVMQMCHL.

Once this process is complete the channel is no longer in doubt. The transmission queue can now be used by another channel, if required.

Problem determination

There are two distinct aspects to problem determination - problems discovered when a command is being submitted, and problems discovered during operation of the channels.

Command validation

Commands and panel data must be free from errors before they are accepted for processing. Any errors found by the validation are immediately notified to the user by error messages.

Problem diagnosis begins with the interpretation of these error messages and taking corrective action.

Processing problems

Problems found during normal operation of the channels are notified to the system console or the system log. Problem diagnosis begins with the collection of all relevant information from the log, and continues with analysis to identify the problem.

Confirmation and error messages are returned to the terminal that initiated the commands, when possible.

IBM MQ produces accounting and statistical data, which you can use to identify trends in utilization and performance. On z/OS, this information is produced in the form of SMF records, see Monitoring performance and resource usage for details. The equivalent information about other platforms is produced as PCF records, see Structure data types for details.

Messages and codes

For messages and codes to help with the primary diagnosis of the problem, see Messages and reason codes.

Safety of messages

In addition to the typical recovery features of IBM MQ, distributed queue management ensures that messages are delivered properly by using a sync point procedure coordinated between the two ends of the message channel. If this procedure detects an error, it closes the channel so that you can investigate the problem, and keeps the messages safely in the transmission queue until the channel is restarted.

The sync point procedure has an added benefit in that it attempts to recover an *in-doubt* situation when the channel starts. (*In-doubt* is the status of a unit of recovery for which a sync point has been requested but the outcome of the request is not yet known.) Also associated with this facility are the two functions:

1. Resolve with commit or backout
2. Reset the sequence number

The use of these functions occurs only in exceptional circumstances because the channel recovers automatically in most cases.

Fast, nonpersistent messages

The nonpersistent message speed (NPMSPEED) channel attribute can be used to specify that any nonpersistent messages on the channel are to be delivered more quickly. For more information about this attribute, see Nonpersistent message speed (NPMSPEED).

If a channel terminates while fast, nonpersistent messages are in transit, the messages might be lost and it is up to the application to arrange for their recovery if required.

If the receiving channel cannot put the message to its destination queue then it is placed on the dead letter queue, if one has been defined. If not, the message is discarded.

Note: If the other end of the channel does not support the option, the channel runs at normal speed.

Undelivered Messages

For information about what happens when a message cannot be delivered, see "What happens when a message cannot be delivered?" on page 165.

What happens when a message cannot be delivered?

When a message cannot be delivered, the MCA can process it in several ways. It can try again, it can return-to-sender, or it can put it on the dead-letter queue.

Figure 23 on page 165 shows the processing that occurs when an MCA is unable to put a message to the destination queue. (The options shown do not apply on all platforms.)

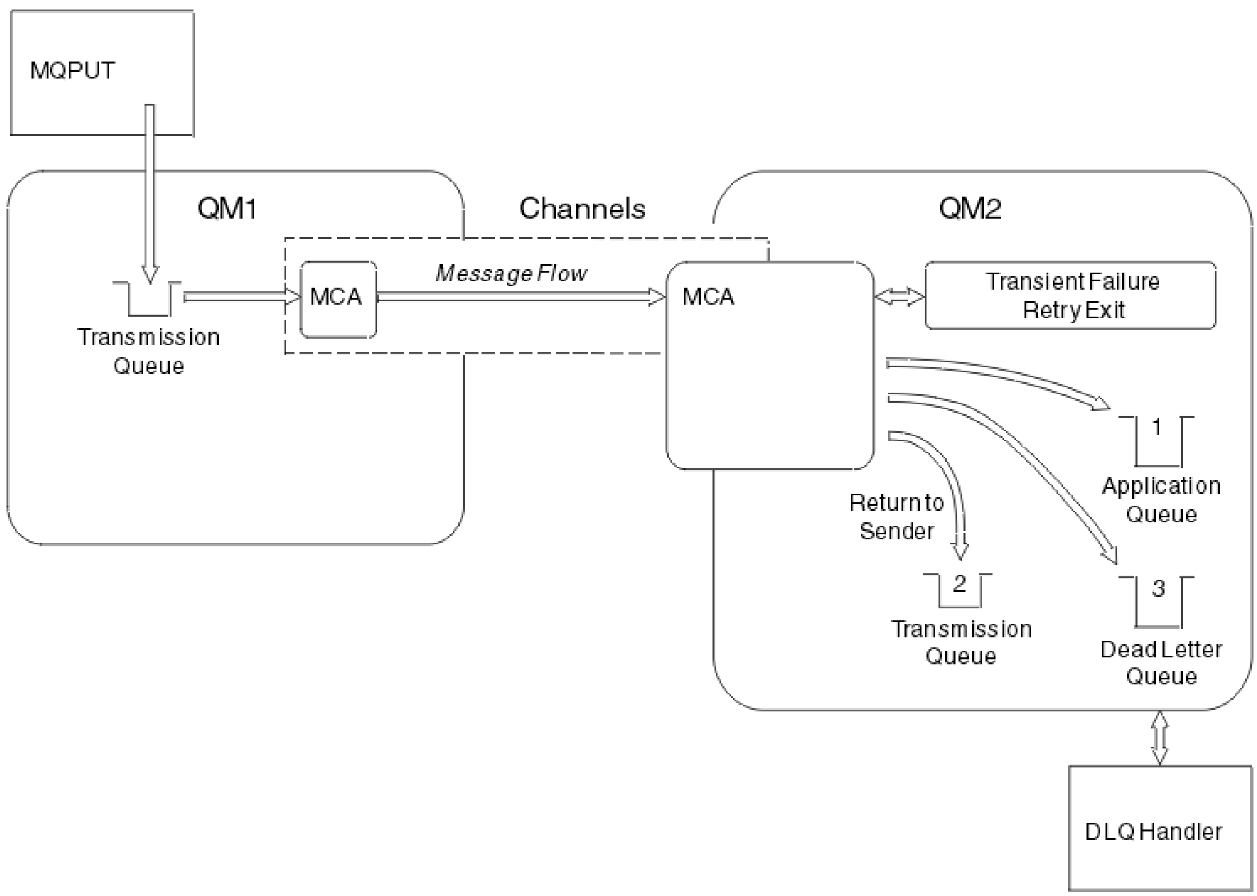

Figure 23. What happens when a message cannot be delivered

As shown in the figure, the MCA can do several things with a message that it cannot deliver. The action taken is determined by options specified when the channel is defined and on the MQPUT report options for the message.

1. Message-retry

   If the MCA is unable to put a message to the target queue for a reason that could be transitory (for example, because the queue is full), the MCA can wait and try the operation again later. You can determine if the MCA waits, for how long, and how many times it tries.

   - You can specify a message-retry time and interval for MQPUT errors when you define your channel. If the message cannot be put to the destination queue because the queue is full, or is inhibited for puts, the MCA tries the operation the number of times specified, at the time interval specified.
   - You can write your own message-retry exit. The exit enables you to specify under what conditions you want the MCA to try the MQPUT or MQOPEN operation again. Specify the name of the exit when you define the channel.
2. Return-to-sender

   If message-retry was unsuccessful, or a different type of error was encountered, the MCA can send the message back to the originator. To enable return-to-sender, you need to specify the following options in the message descriptor when you put the message to the original queue:

   - The MQRO_EXCEPTION_WITH_FULL_DATA report option
   - The MQRO_DISCARD_MSG report option
   - The name of the reply-to queue and reply-to queue manager

   If the MCA is unable to put the message to the destination queue, it generates an exception report containing the original message, and puts it on a transmission queue to be sent to the reply-to queue specified in the original message. (If the reply-to queue is on the same queue manager as the MCA, the message is put directly to that queue, not to a transmission queue.)
3. Dead-letter queue

   If a message cannot be delivered or returned, it is put on to the dead-letter queue (DLQ). You can use the DLQ handler to process the message. This processing is described in Processing messages on a dead-letter queue for IBM MQ for UNIX, Linux and Windows systems, and in The dead-letter queue handler utility (CSQUDLQH) for z/OS systems. If the dead-letter queue is not available, the sending MCA leaves the message on the transmission queue, and the channel stops. On a fast channel, nonpersistent messages that cannot be written to a dead-letter queue are lost.

   On IBM WebSphere MQ 7.0, if no local dead-letter queue is defined, the remote queue is not available or defined, and there is no remote dead-letter queue, then the sender channel goes into RETRY and messages are automatically rolled back to the transmission queue.

Related reference

Use Dead-Letter Queue (USEDLQ)

Triggering channels

IBM MQ provides a facility for starting an application automatically when certain conditions on a queue are met. This facility is called triggering.

This explanation is intended as an overview of triggering concepts. For a complete description, see Starting IBM MQ applications using triggers.

For platform-specific information see the following:

- For Windows, see UNIX and Linux systems, "Triggering channels on UNIX, Linux and Windows systems." on page 168
- IBM i For IBM i, see "Triggering channels in IBM MQ for IBM i" on page 168
- z/OS For z/OS, see "Transmission queues and triggering channels" on page 590

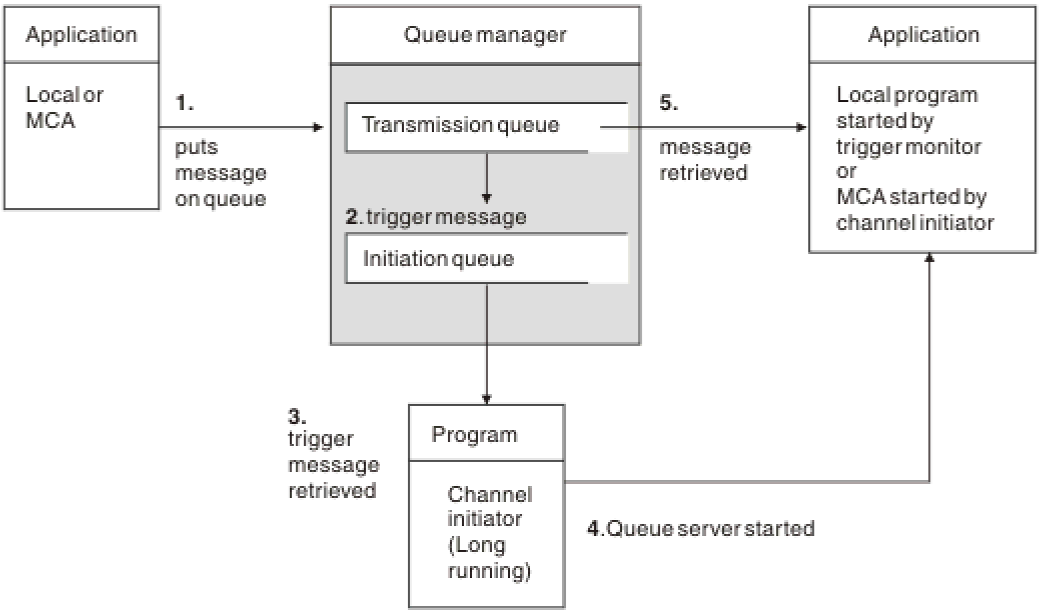

Figure 24. The concepts of triggering

The objects required for triggering are shown in Figure 24 on page 167. It shows the following sequence of events:

1. The local queue manager places a message from an application or from a message channel agent (MCA) on the transmission queue.
2. When the triggering conditions are fulfilled, the local queue manager places a trigger message on the initiation queue.
3. The long-running channel initiator program monitors the initiation queue, and retrieves messages as they arrive.
4. The channel initiator processes the trigger messages according to information contained in them. This information might include the channel name, in which case the corresponding MCA is started.
5. The local application or the MCA, having been triggered, retrieves the messages from the transmission queue.

To set up this scenario, you need to:

- Create the transmission queue with the name of the initiation queue (that is, SYSTEM.CHANNEL.INITQ) in the corresponding attribute.
- Ensure that the initiation queue (SYSTEM.CHANNEL.INITQ) exists.
- Ensure that the channel initiator program is available and running. The channel initiator program must be provided with the name of the initiation queue in its start command. z/OS On z/OS, the name of the initiation queue is fixed, so is not used on the start command.
- Optionally, create the process definition for the triggering, if it does not exist, and ensure that the *UserData* field contains the name of the channel it serves. Instead of creating a process definition, you can specify the channel name in the *TriggerData* attribute of the transmission queue. IBM MQ for IBM i IBM i, UNIX, Linux and Windows systems, allow the channel name to be specified as blank, in which case the first available channel definition with this transmission queue is used.
- Ensure that the transmission queue definition contains the name of the process definition to serve it, (if applicable), the initiation queue name, and the triggering characteristics you feel are most suitable. The trigger control attribute allows triggering to be enabled, or not, as necessary.

Note:

1. The channel initiator program acts as a 'trigger monitor' monitoring the initiation queue used to start channels.
2. An initiation queue and trigger process can be used to trigger any number of channels.
3. Any number of initiation queues and trigger processes can be defined.
4. A trigger type of FIRST is recommended, to avoid flooding the system with channel starts.

Triggering channels on UNIX, Linux and Windows systems.

You can create a process definition in IBM MQ, defining processes to be triggered. Use the MQSC command DEFINE PROCESS to create a process definition naming the process to be triggered when messages arrive on a transmission queue. The USERDATA attribute of the process definition contains the name of the channel being served by the transmission queue.

Define the local queue (QM4), specifying that trigger messages are to be written to the initiation queue (IQ) to trigger the application that starts channel (QM3.TO.QM4):

```
DEFINE QLOCAL(QM4) TRIGGER INITQ(SYSTEM.CHANNEL.INITQ) PROCESS(P1) USAGE(XMITQ)
```

Define the application (process P1) to be started:

```
DEFINE PROCESS(P1) USERDATA(QM3.TO.QM4)
```

Alternatively, for IBM MQ for UNIX, Linux and Windows systems, you can eliminate the need for a process definition by specifying the channel name in the TRIGDATA attribute of the transmission queue.

Define the local queue (QM4). Specify that trigger messages are to be written to the default initiation queue SYSTEM.CHANNEL.INITQ, to trigger the application (process P1) that starts channel (QM3.TO.QM4):

```
DEFINE QLOCAL(QM4) TRIGGER INITQ(SYSTEM.CHANNEL.INITQ)
USAGE(XMITQ) TRIGDATA(QM3.TO.QM4)
```

If you do not specify a channel name, the channel initiator searches the channel definition files until it finds a channel that is associated with the named transmission queue.

Triggering channels in IBM MQ for IBM i

IBM i

Triggering of channels in IBM MQ for IBM i is implemented with the channel initiator process. A channel initiator process for the initiation queue SYSTEM.CHANNEL.INITQ is started automatically with the queue manager unless it is disabled by altering the queue manager SCHINIT attribute.

Set up the transmission queue for the channel, specifying SYSTEM.CHANNEL.INITQ as the initiation queue, and enabling triggering for the queue. The channel initiator starts the first available channel that specifies this transmission queue.

```
CRTMQMQ QNAME(MYXMITQ1) QTYPE(*LCL) MQMNAME(MYQMGR)
TRGENBL(*YES) INITQNAME(SYSTEM.CHANNEL.INITQ)
USAGE(*TMQ)
```

You can manually start up to three channel initiator processes with the STRMQMCHLI command and specify different initiation queues. You can also specify more than one channel able to process the transmission queue and choose which channel to start. This capability is still provided to be compatible with earlier releases. Its usage is deprecated.

Note: Only one channel at a time can process a transmission queue.

```
STRMQMCHLI QNAME(MYINITQ)
```

Set up the transmission queue for the channel, specifying TRGENBL(*YES) and, to choose which channel to attempt to start, specify the channel name in the TRIGDATA field. For example:

```
CRTMQMQ QNAME(MYXMITQ2) QTYPE(*LCL) MQMNAME(MYQMGR)
TRGENBL(*YES) INITQNAME(MYINITQ)
USAGE(*TMQ) TRIGDATA(MYCHANNEL)
```

Related concepts

"Starting and stopping the channel initiator" on page 169
Triggering is implemented using the channel initiator process.

"Configuring distributed queuing" on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Related reference

Channel programs on UNIX, Linux, and Windows systems

IBM i Intercommunication jobs on IBM i

IBM i Channel states on IBM i

Starting and stopping the channel initiator

Triggering is implemented using the channel initiator process.

This channel initiator process is started with the MQSC command START CHINIT. Unless you are using the default initiation queue, specify the name of the initiation queue on the command. For example, to use the START CHINIT command to start queue IQ for the default queue manager, enter:

```
START CHINIT INITQ(IQ)
```

By default, a channel initiator is started automatically using the default initiation queue, SYSTEM.CHANNEL.INITQ. If you want to start all your channel initiators manually, follow these steps:

1. Create and start the queue manager.
2. Alter the queue manager's SCHINIT property to MANUAL
3. End and restart the queue manager

In IBM i IBM MQ for iSeries, UNIX, Linux and Windows systems, a channel initiator is started automatically. The number of channel initiators that you can start is limited. The default and maximum value is 3. You can change this using MAXINITIATORS in the qm.ini file for UNIX and Linux systems, and in the registry for Windows systems.

See IBM MQ Control commands for details of the run channel initiator command **runmqchi**, and the other control commands.

Stopping the channel initiator

The default channel initiator is started automatically when you start a queue manager. All channel initiators are stopped automatically when a queue manager is stopped.

Initialization and configuration files

The handling of channel initialization data depends on your IBM MQ platform.

z/OS systems

z/OS

In IBM MQ for z/OS, initialization and configuration information is specified using the ALTER QMGR MQSC command. If you put ALTER QMGR commands in the CSQINP2 initialization input data set, they are processed every time the queue manager is started.

To run MQSC commands such as START LISTENER every time you start the channel initiator, put them in the CSQINPX initialization input data set and specify the optional DD statement CSQINPX in the channel initiator started task procedure.

For further information about CSQINP2 and CSQINPX, see Customize the initialization input data sets, and ALTER QMGR.

Windows, IBM i IBM i, UNIX and Linux systems

In IBM MQ for Windows, IBM i IBM i, UNIX and Linux systems, there are *configuration files* to hold basic configuration information about the IBM MQ installation.

There are two configuration files: one applies to the machine, the other applies to an individual queue manager.

IBM MQ configuration file

This file holds information relevant to all the queue managers on the IBM MQ system. The file is called mqs.ini. It is fully described in the Administering for IBM MQ for Windows, IBM i and in Administering IBM i , and in UNIX and Linux systems.

Queue manager configuration file

This file holds configuration information relating to one particular queue manager. The file is called `qm.ini`.

It is created during queue manager creation and can hold configuration information relevant to any aspect of the queue manager. Information held in the file includes details of how the configuration of the log differs from the default in the IBM MQ configuration file.

The queue manager configuration file is held in the root of the directory tree occupied by the queue manager. For example, for the DefaultPath attributes, the queue manager configuration files for a queue manager called QMNAME would be:

For UNIX and Linux systems:

```
/var/mqm/qmgrs/QMNAME/qm.ini
```

An excerpt of a qm.ini file follows. It specifies that the TCP/IP listener is to listen on port 2500, the maximum number of current channels is to be 200, and the maximum number of active channels is to be 100.

```
TCP:
Port=2500
CHANNELS:
MaxChannels=200
MaxActiveChannels=100
```

You can specify a range of TCP/IP ports to be used by an outbound channel. One method is to use the qm.ini file to specify the start and end of a range of port values. The following example shows a qm.ini file specifying a range of channels:

```
TCP:
StrPort=2500
EndPort=3000
CHANNELS:
MaxChannels=200
MaxActiveChannels=100
```

If you specify a value for StrPort or EndPort then you must specify a value for both. The value of EndPort must always be greater than the value of StrPort.

The channel tries to use each of the port values in the range specified. When the connection is successful, the port value is the port that the channel then uses.

IBM i For IBM i:

```
/QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini
```

For Windows systems:

```
C:\ProgramData\IBM\MQ\qmgrs\QMNAME\qm.ini
```

For more information about qm.ini files, see Configuration file stanzas for distributed queuing.

Data conversion for messages

IBM MQ messages might require data conversion when sent between queues on different queue managers.

An IBM MQ message consists of two parts:

- Control information in a message descriptor
- Application data

Either of the two parts might require data conversion when sent between queues on different queue managers. For information about application data conversion, see Application data conversion.

Writing your own message channel agents

IBM MQ allows you to write your own message channel agent (MCA) programs or to install one from an independent software vendor.

You might want to write your own MCA programs to make IBM MQ interoperate over your own proprietary communications protocol, or to send messages over a protocol that IBM MQ does not support. (You cannot write your own MCA to interoperate with an IBM MQ-supplied MCA at the other end.)

If you decide to use an MCA that was not supplied by IBM MQ, you must consider the following points.

Message sending and receiving

You must write a sending application that gets messages from wherever your application puts them, for example from a transmission queue, and sends them out on a protocol with which you want to communicate. You must also write a receiving application that takes messages from this protocol and puts them onto destination queues. The sending and receiving applications use the message queue interface (MQI) calls, not any special interfaces.

You must ensure that messages are only delivered once. Sync point coordination can be used to help with this delivery.

Channel control function

You must provide your own administration functions to control channels. You cannot use IBM MQ channel administration functions either for configuring (for example, the DEFINE CHANNEL command) or monitoring (for example, DISPLAY CHSTATUS) your channels.

Initialization file
You must provide your own initialization file, if you require one.

Application data conversion
You probably want to allow for data conversion for messages you send to a different system. If so, use the MQGMO_CONVERT option on the MQGET call when retrieving messages from wherever your application puts them, for example the transmission queue.

User exits
Consider whether you need user exits. If so, you can use the same interface definitions that IBM MQ uses.

Triggering
If your application puts messages to a transmission queue, you can set up the transmission queue attributes so that your sending MCA is triggered when messages arrive on the queue.

Channel initiator
You might must provide your own channel initiator.

Other things to consider for distributed queue management

Other topics to consider when preparing IBM MQ for distributed queue management. This topic covers Undelivered-message queue, Queues in use, System extensions and user-exit programs, and Running channels and listeners as trusted applications.

Undelivered-message queue

To ensure that messages arriving on the undelivered-message queue (also known as the dead-letter queue or DLQ) are processed, create a program that can be triggered or run at regular intervals to handle these messages. A DLQ handler is provided with IBM MQ on UNIX and Linux systems; for more information, see The sample DLQ handler, amqsdlq. IBM i For more information on IBM MQ for IBM i, see The IBM MQ for IBM i dead-letter queue handler.

Queues in use

MCAs for receiver channels can keep the destination queues open even when messages are not being transmitted. This results in the queues appearing to be "in use".

Maximum number of channels

IBM i On IBM MQ for IBM i you can specify the maximum number of channels allowed in your system and the maximum number that can be active at one time. You specify these numbers in the qm.ini file in directory QIBM/UserData/mqm/qmgrs/*queue_manager_name*. See Configuration file stanzas for distributed queuing.

System extensions and user-exit programs

A facility is provided in the channel definition to enable extra programs to be run at defined times during the processing of messages. These programs are not supplied with IBM MQ, but can be provided by each installation according to local requirements.

In order to run, these user-exit programs must have predefined names and be available on call to the channel programs. The names of the user-exit programs are included in the message channel definitions.

There is a defined control block interface for handing over control to these programs, and for handling the return of control from these programs.

The precise places where these programs are called, and details of control blocks and names, are to be found in Channel-exit programs for messaging channels.

Running channels and listeners as trusted applications

If performance is an important consideration in your environment and your environment is stable, you can run your channels and listeners as trusted, using the FASTPATH binding. There are two factors that influence whether channels and listeners run as trusted:

- The environment variable MQ_CONNECT_TYPE=FASTPATH or MQ_CONNECT_TYPE=STANDARD. This is case-sensitive. If you specify a value that is not valid it is ignored.
- MQIBindType in the Channels stanza of the `qm.ini` or registry file. You can set this to FASTPATH or STANDARD and it is not case-sensitive. The default is STANDARD.

You can use MQIBindType in association with the environment variable to achieve the required effect as follows:

MQIBindType	Environment variable	Result
STANDARD	UNDEFINED	STANDARD
FASTPATH	UNDEFINED	FASTPATH
STANDARD	STANDARD	STANDARD
FASTPATH	STANDARD	STANDARD
STANDARD	FASTPATH	STANDARD
FASTPATH	FASTPATH	FASTPATH
STANDARD	CLIENT	CLIENT
FASTPATH	CLIENT	STANDARD
STANDARD	LOCAL	STANDARD
FASTPATH	LOCAL	STANDARD

In summary, there are only two ways of actually making channels and listeners run as trusted:

1. By specifying MQIBindType=FASTPATH in `qm.ini` or registry and not specifying the environment variable.
2. By specifying MQIBindType=FASTPATH in `qm.ini` or registry and setting the environment variable to FASTPATH.

Consider running listeners as trusted, because listeners are stable processes. Consider running channels as trusted, unless you are using unstable channel exits or the command STOP CHANNEL MODE(TERMINATE).

Monitoring and controlling channels on Windows, UNIX and Linux platforms

For DQM you need to create, monitor, and control the channels to remote queue managers. You can control channels using commands, programs, IBM MQ Explorer, files for the channel definitions, and a storage area for synchronization information.

You can use the following types of command:

The IBM MQ commands (MQSC)

You can use the MQSC as single commands in an MQSC session in Windows, UNIX and Linux systems. To issue more complicated, or multiple, commands the MQSC can be built into a file that you then run from the command line. For details, see MQSC commands. This section gives some simple examples of using MQSC for distributed queuing.

The channel commands are a subset of the IBM MQ Commands (MQSC). You use MQSC and the control commands to:

- Create, copy, display, change, and delete channel definitions

- Start and stop channels, ping, reset channel sequence numbers, and resolve in-doubt messages when links cannot be re-established
- Display status information about channels

Control commands
: You can also issue *control commands* at the command line for some of these functions. For details, see control commands.

Programmable command format commands
: For details, see PCF commands.

IBM MQ Explorer
: On UNIX, Linux and Windows systems, you can use the IBM MQ Explorer. This provides a graphical administration interface to perform administrative tasks as an alternative to using control commands or MQSC commands. Channel definitions are held as queue manager objects.

Each queue manager has a DQM component for controlling interconnections to compatible remote queue managers. A storage area holds sequence numbers and *logical unit of work (LUW)* identifiers. These are used for channel synchronization purposes.

For a list of the functions available to you when setting up and controlling message channels, using the different types of command, see Table 20 on page 175.

Related concepts

"Getting started with objects" on page 176
Channels must be defined, and their associated objects must exist and be available for use, before a channel can be started. This section shows you how.

"Setting up communication for Windows" on page 183
When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This section explains how to do this using one of the four forms of communication for IBM MQ for Windows systems.

"Setting up communication on UNIX and Linux systems" on page 189
DQM is a remote queuing facility for IBM MQ. It provides channel control programs for the queue manager which form the interface to communication links, controllable by the system operator. The channel definitions held by distributed-queuing management use these connections.

"Monitoring and controlling channels on IBM i" on page 195
Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers. Each queue manager has a DQM program for controlling interconnections to compatible remote queue managers.

Related reference

Channel programs on UNIX, Linux, and Windows systems

Message channel planning example for distributed platforms

Example configuration information

Channel attributes

Functions required for setting up and controlling channels

A number of IBM MQ functions might be needed to set up and control channels. The channel functions are explained in this topic.

You can create a channel definition using the default values supplied by IBM MQ, specifying the name of the channel, the type of channel you are creating, the communication method to be used, the transmission queue name and the connection name.

The channel name must be the same at both ends of the channel, and unique within the network. However, you must restrict the characters used to those that are valid for IBM MQ object names.

For other channel related functions, see the following topics:

Table 20 on page 175 shows the full list of IBM MQ functions that you might need.

Table 20. Functions required in Windows, UNIX and Linux systems

Function	**Control commands**	**MQSC**	**IBM MQ Explorer equivalent?**
Queue manager functions			
Change queue manager		ALTER QMGR	Yes
Create queue manager	crtmqm		Yes
Delete queue manager	dltmqm		Yes
Display queue manager		DISPLAY QMGR	Yes
End queue manager	endmqm		Yes
Ping queue manager		PING QMGR	No
Start queue manager	strmqm		Yes
Command server functions			
Display command server	dspmqcsv		No
End command server	endmqcsv		No
Start command server	strmqcsv		No
Queue functions			
Change queue		ALTER QALIAS ALTER QLOCAL ALTER QMODEL ALTER QREMOTE See, ALTER queues.	Yes
Clear queue		CLEAR QLOCAL	Yes
Create queue		DEFINE QALIAS DEFINE QLOCAL DEFINE QMODEL DEFINE QREMOTE See, DEFINE queues.	Yes

Table 20. Functions required in Windows, UNIX and Linux systems (continued)			
Function	**Control commands**	**MQSC**	**IBM MQ Explorer equivalent?**
Delete queue		DELETE QALIAS DELETE QLOCAL DELETE QMODEL DELETE QREMOTE See, DELETE queues.	Yes
Display queue		DISPLAY QUEUE	Yes
Process functions			
Change process		ALTER PROCESS	Yes
Create process		DEFINE PROCESS	Yes
Delete process		DELETE PROCESS	Yes
Display process		DISPLAY PROCESS	Yes
Channel functions			
Change channel		ALTER CHANNEL	Yes
Create channel		DEFINE CHANNEL	Yes
Delete channel		DELETE CHANNEL	Yes
Display channel		DISPLAY CHANNEL	Yes
Display channel status		DISPLAY CHSTATUS	Yes
End channel		STOP CHANNEL	Yes
Ping channel		PING CHANNEL	Yes
Reset channel		RESET CHANNEL	Yes
Resolve channel		RESOLVE CHANNEL	Yes
Run channel	runmqchl	START CHANNEL	Yes
Run channel initiator	runmqchi	START CHINIT	No
Run listener [1]	runmqlsr	START LISTENER	No
End listener	endmqlsr (Windows systems, AIX, HP-UX, and Solaris only)		No

Note:

1. A listener might be started automatically when the queue manager starts.

Getting started with objects

Channels must be defined, and their associated objects must exist and be available for use, before a channel can be started. This section shows you how.

Use the IBM MQ commands (MQSC) or the IBM MQ Explorer to:

1. Define message channels and associated objects
2. Monitor and control message channels

The associated objects you might need to define are:

- Transmission queues
- Remote queue definitions
- Queue manager alias definitions
- Reply-to queue alias definitions
- Reply-to local queues
- Processes for triggering (MCAs)
- Message channel definitions

The particular communication link for each channel must be defined and available before a channel can be run. For a description of how LU 6.2, TCP/IP, NetBIOS, SPX, and DECnet links are defined, see the particular communication guide for your installation. See also Example configuration information.

For more information about creating and working with objects, see the following subtopics:

Creating associated objects

MQSC is used to create associated objects.

Use MQSC to create the queue and alias objects: transmission queues, remote queue definitions, queue manager alias definitions, reply-to queue alias definitions, and reply-to local queues.

Also create the definitions of processes for triggering (MCAs) in a similar way.

For an example showing how to create all the required objects see Message channel planning example for distributed platforms.

Creating default objects

Default objects are created automatically when a queue manager is created. These objects are queues, channels, a process definition, and administration queues. After the default objects have been created, you can replace them at any time by running the strmqm command with the -c option.

When you use the crtmqm command to create a queue manager, the command also initiates a program to create a set of default objects.

1. Each default object is created in turn. The program keeps a count of how many objects are successfully defined, how many existed and were replaced, and how many unsuccessful attempts there were.
2. The program displays the results to you and if any errors occurred, directs you to the appropriate error log for details.

When the program has finished running, you can use the strmqm command to start the queue manager.

See The control commands for more information about the crtmqm and strmqm commands.

Changing the default objects

When you specify the -c option, the queue manager is started temporarily while the objects are created and is then shut down again. Issuing strmqm with the -c option refreshes existing system objects with the default values (for example, the MCAUSER attribute of a channel definition is set to blanks). You must use the strmqm command again, without the -c option, if you want to start the queue manager.

If you want to change the default objects, you can create your own version of the old amqscoma.tst file and edit it.

Creating a channel

Create ***two*** channel definitions, one at each end of the connection. You create the first channel definition at the first queue manager. Then you create the second channel definition at the second queue manager, on the other end of the link.

Both ends must be defined using the ***same*** channel name. The two ends must have **compatible** channel types, for example: Sender and Receiver.

To create a channel definition for one end of the link use the MQSC command DEFINE CHANNEL. Include the name of the channel, the channel type for this end of the connection, a connection name, a description (if required), the name of the transmission queue (if required), and the transmission protocol. Also include any other attributes that you want to be different from the system default values for the required channel type, using the information you have gathered previously.

You are provided with help in deciding on the values of the channel attributes in Channel attributes.

Note: You are recommended to name all the channels in your network uniquely. Including the source and target queue manager names in the channel name is a good way to do this.

Create channel example

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) +
DESCR('Sender channel to QM2') +
CONNAME(QM2) TRPTYPE(TCP) XMITQ(QM2) CONVERT(YES)
```

In all the examples of MQSC the command is shown as it appears in a file of commands, and as it is typed in Windows or UNIX or Linux systems. The two methods look identical, except that to issue a command interactively, you must first start an MQSC session. Type `runmqsc`, for the default queue manager, or `runmqsc` *qmname* where *qmname* is the name of the required queue manager. Then type any number of commands, as shown in the examples.

For portability, restrict the line length of your commands to 72 characters. Use the concatenation character, + , as shown to continue over more than one line. On Windows use Ctrl-z to end the entry at the command line. On UNIX and Linux systems, use Ctrl-d. Alternatively, on UNIX, Linux or Windows systems, use the **end** command.

Displaying a channel

Use the MQSC command DISPLAY CHANNEL to display the attributes of a channel.

The ALL parameter of the DISPLAY CHANNEL command is assumed by default if no specific attributes are requested and the channel name specified is not generic.

The attributes are described in Channel attributes.

Display channel examples

```
DISPLAY CHANNEL(QM1.TO.QM2) TRPTYPE,CONVERT

DISPLAY CHANNEL(QM1.TO.*) TRPTYPE,CONVERT

DISPLAY CHANNEL(*) TRPTYPE,CONVERT

DISPLAY CHANNEL(QM1.TO.QMR34) ALL
```

Displaying channel status

Use the MQSC command DISPLAY CHSTATUS, specifying the channel name and whether you want the current status of channels or the status of saved information.

DISPLAY CHSTATUS applies to all message channels. It does not apply to MQI channels other than server-connection channels.

Information displayed includes:

- Channel name
- Communication connection name
- In-doubt status of channel (where appropriate)
- Last sequence number
- Transmission queue name (where appropriate)

- The in-doubt identifier (where appropriate)
- The last committed sequence number
- Logical unit of work identifier
- Process ID
- Thread ID (Windows only)

Display channel status examples

```
DISPLAY CHSTATUS(*) CURRENT

DISPLAY CHSTATUS(QM1.TO.*) SAVED
```

The saved status does not apply until at least one batch of messages has been transmitted on the channel. Status is also saved when a channel is stopped (using the STOP CHL command) and when the queue manager is ended.

Checking links using Ping

Use the MQSC command PING CHANNEL to exchange a fixed data message with the remote end.

Ping gives some confidence to the system supervisor that the link is available and functioning.

Ping does not involve the use of transmission queues and target queues. It uses channel definitions, the related communication link, and the network setup. It can only be used if the channel is not currently active.

It is available from sender, server, and cluster-sender channels only. The corresponding channel is started at the far side of the link, and performs the startup parameter negotiation. Errors are notified normally.

The result of the message exchange is presented as `Ping complete` or an error message.

Ping with LU 6.2

When Ping is invoked, by default no user ID or password flows to the receiving end. If user ID and password are required, they can be created at the initiating end in the channel definition. If a password is entered into the channel definition, it is encrypted by IBM MQ before being saved. It is then decrypted before flowing across the conversation.

Starting a channel

Use the MQSC command START CHANNEL for sender, server, and requester channels. For applications to be able to exchange messages, you must start a listener program for inbound connections.

START CHANNEL is not necessary where a channel has been set up with queue manager triggering.

When started, the sending MCA reads the channel definitions and opens the transmission queue. A channel start-up sequence is issued, which remotely starts the corresponding MCA of the receiver or server channel. When they have been started, the sender and server processes await messages arriving on the transmission queue and transmit them as they arrive.

When you use triggering or run channels as threads, ensure that the channel initiator is available to monitor the initiation queue. The channel initiator is started by default as part of the queue manager.

However, TCP and LU 6.2 do provide other capabilities:

- For TCP on UNIX and Linux systems, inetd can be configured to start a channel. inetd is started as a separate process.
- For LU 6.2 in UNIX and Linux systems, configure your SNA product to start the LU 6.2 responder process.

- For LU 6.2 in Windows systems, using SNA Server you can use TpStart (a utility provided with SNA Server) to start a channel. TpStart is started as a separate process.

Use of the Start option always causes the channel to resynchronize, where necessary.

For the start to succeed:

- Channel definitions, local and remote, must exist. If there is no appropriate channel definition for a receiver or server-connection channel, a default one is created automatically if the channel is auto-defined. See Channel auto-definition exit program.
- Transmission queue must exist, and have no other channels using it.
- MCAs, local and remote, must exist.
- Communication link must be available.
- Queue managers must be running, local and remote.
- Message channel must not be already running.

A message is returned to the screen confirming that the request to start a channel has been accepted. For confirmation that the start command has succeeded, check the error log, or use DISPLAY CHSTATUS. The error logs are:

Windows

MQ_INSTALLATION_PATH`\qmgrs\qmname\errors\AMQERR01.LOG` (for each queue manager called qmname)

MQ_INSTALLATION_PATH`\qmgrs\@SYSTEM\errors\AMQERR01.LOG` (for general errors)

MQ_INSTALLATION_PATH represents the high-level directory in which IBM MQ is installed.

Note: On Windows systems, you still also get a message in the Windows systems application event log.

UNIX and Linux systems

`/var/mqm/qmgrs/qmname/errors/AMQERR01.LOG` (for each queue manager called qmname)

`/var/mqm/qmgrs/@SYSTEM/errors/AMQERR01.LOG` (for general errors)

On Windows, UNIX and Linux systems, use the runmqlsr command to start the IBM MQ listener process. By default, any inbound requests for channel attachment causes the listener process to start MCAs as threads of the amqrmppa process.

```
runmqlsr -t tcp -m QM2
```

For outbound connections, you must start the channel in one of the following three ways:

1. Use the MQSC command START CHANNEL, specifying the channel name, to start the channel as a process or a thread, depending on the MCATYPE parameter. (If channels are started as threads, they are threads of a channel initiator.)

   ```
   START CHANNEL(QM1.TO.QM2)
   ```

2. Use the control command runmqchl to start the channel as a process.

   ```
   runmqchl -c QM1.TO.QM2 -m QM1
   ```

3. Use the channel initiator to trigger the channel.

Stopping a channel

Use the MQSC command STOP CHANNEL to request the channel to stop activity. The channel does not start a new batch of messages until the operator starts the channel again.

For information about restarting stopped channels, see “Restarting stopped channels” on page 162.

This command can be issued to a channel of any type except MQCHT_CLNTCONN.

You can select the type of stop you require:

Stop quiesce example

```
STOP CHANNEL(QM1.TO.QM2) MODE(QUIESCE)
```

This command requests the channel to close down in an orderly way. The current batch of messages is completed and the sync point procedure is carried out with the other end of the channel. If the channel is idle this command does not terminate a receiving channel.

Stop force example

```
STOP CHANNEL(QM1.TO.QM2) MODE(FORCE)
```

This option stops the channel immediately, but does not terminate the channel's thread or process. The channel does not complete processing the current batch of messages, and can, therefore, leave the channel in doubt. In general, consider using the quiesce stop option.

Stop terminate example

```
STOP CHANNEL(QM1.TO.QM2) MODE(TERMINATE)
```

This option stops the channel immediately, and terminates the channel's thread or process.

Stop (quiesce) stopped example

```
STOP CHANNEL(QM1.TO.QM2) STATUS(STOPPED)
```

This command does not specify a MODE, so defaults to MODE(QUIESCE). It requests that the channel is stopped so that it cannot be restarted automatically but must be started manually.

Stop (quiesce) inactive example

```
STOP CHANNEL(QM1.TO.QM2) STATUS(INACTIVE)
```

This command does not specify a MODE, so defaults to MODE(QUIESCE). It requests that the channel is made inactive so that it restarts automatically when required.

Renaming a channel

Use MQSC to rename a message channel.

Use MQSC to carry out the following steps:

1. Use STOP CHANNEL to stop the channel.
2. Use DEFINE CHANNEL to create a duplicate channel definition with the new name.
3. Use DISPLAY CHANNEL to check that it has been created correctly.
4. Use DELETE CHANNEL to delete the original channel definition.

If you decide to rename a message channel, remember that a channel has ***two*** channel definitions, one at each end. Make sure that you rename the channel at both ends at the same time.

Resetting a channel

Use the MQSC command RESET CHANNEL to change the message sequence number.

The RESET CHANNEL command is available for any message channel, but not for MQI channels (client-connection or server-connection). The first message starts the new sequence the next time the channel is started.

If the command is issued on a sender or server channel, it informs the other side of the change when the channel is restarted.

Related concepts

"Getting started with objects" on page 176
Channels must be defined, and their associated objects must exist and be available for use, before a channel can be started. This section shows you how.

"Channel control function" on page 152
The channel control function provides facilities for you to define, monitor, and control channels.

"Configuring distributed queuing" on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Related reference

RESET CHANNEL

Resolving in-doubt messages on a channel

Use the MQSC command RESOLVE CHANNEL when messages are held in-doubt by a sender or server. For example because one end of the link has terminated, and there is no prospect of it recovering.

The RESOLVE CHANNEL command accepts one of two parameters: BACKOUT or COMMIT. Backout restores messages to the transmission queue, while Commit discards them.

The channel program does not try to establish a session with a partner. Instead, it determines the logical unit of work identifier (LUWID) which represents the in-doubt messages. It then issues, as requested, either:

- BACKOUT to restore the messages to the transmission queue; or
- COMMIT to delete the messages from the transmission queue.

For the resolution to succeed:

- The channel must be inactive
- The channel must be in doubt
- The channel type must be sender, server, or cluster-sender
- A local channel definition must exist
- The local queue manager must be running

Related concepts

"Getting started with objects" on page 176
Channels must be defined, and their associated objects must exist and be available for use, before a channel can be started. This section shows you how.

"Channel control function" on page 152
The channel control function provides facilities for you to define, monitor, and control channels.

"Configuring distributed queuing" on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Related reference

RESOLVE CHANNEL

Setting up communication for Windows

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, it is necessary for the connection to be defined and available. This section explains how to do this using one of the four forms of communication for IBM MQ for Windows systems.

You might find it helpful to refer to Example configuration - IBM MQ for Windows .

For UNIX and Linux systems see “Setting up communication on UNIX and Linux systems” on page 189.

Deciding on a connection

Choose from the following four forms of communication for IBM MQ for Windows systems:

- “Defining a TCP connection on Windows” on page 183
- “Defining an LU 6.2 connection on Windows” on page 185
- “Defining a NetBIOS connection on Windows” on page 187

Each channel definition must specify only one protocol as the Transmission protocol (Transport Type) attribute. One or more protocols can be used by a queue manager.

For IBM MQ clients, it might be useful to have alternative channels using different transmission protocols. For more information about IBM MQ clients, see Overview of clients.

Related concepts

“Configuring distributed queuing” on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

“Monitoring and controlling channels on Windows, UNIX and Linux platforms” on page 173
For DQM you need to create, monitor, and control the channels to remote queue managers. You can control channels using commands, programs, IBM MQ Explorer, files for the channel definitions, and a storage area for synchronization information.

“Configuring connections between the client and server” on page 14
To configure the communication links between IBM MQ MQI clients and servers, decide on your communication protocol, define the connections at both ends of the link, start a listener, and define channels.

Defining a TCP connection on Windows

Define a TCP connection by configuring a channel at the sending end to specify the address of the target, and by running a listener program at the receiving end.

Sending end

Specify the host name, or the TCP address of the target machine, in the Connection name field of the channel definition.

The port to connect to defaults to 1414. Port number 1414 is assigned by the Internet Assigned Numbers Authority to IBM MQ.

To use a port number other than the default, specify it in the connection name field of the channel object definition thus:

```
DEFINE CHANNEL('channel name') CHLTYPE(SDR) +
          TRPTYPE(TCP) +
          CONNAME('OS2ROG3(1822)') +
          XMITQ('XMitQ name') +
          REPLACE
```

where 0S2R0G3 is the DNS name of the remote queue manager and 1822 is the port required. (This must be the port that the listener at the receiving end is listening on.)

A running channel must be stopped and restarted to pick up any change to the channel object definition.

You can change the default port number by specifying it in the `.ini` file for IBM MQ for Windows:

```
TCP:
Port=1822
```

Note: To select which TCP/IP port number to use, IBM MQ uses the first port number it finds in the following sequence:

1. The port number explicitly specified in the channel definition or command line. This number allows the default port number to be overridden for a channel.
2. The port attribute specified in the TCP stanza of the `.ini` file. This number allows the default port number to be overridden for a queue manager.
3. The default value of 1414. This is the number assigned to IBM MQ by the Internet Assigned Numbers Authority for both inbound and outbound connections.

For more information about the values you set using qm.ini, see Configuration file stanzas for distributed queuing.

Receiving on TCP

To start a receiving channel program, a listener program must be started to detect incoming network requests and start the associated channel. You can use the IBM MQ listener.

Receiving channel programs are started in response to a startup request from the sending channel.

To start a receiving channel program, a listener program must be started to detect incoming network requests and start the associated channel. You can use the IBM MQ listener.

To run the Listener supplied with IBM MQ, that starts new channels as threads, use the runmqlsr command.

A basic example of using the **`runmqlsr`** command:

```
runmqlsr -t tcp [-m QMNAME] [-p 1822]
```

The square brackets indicate optional parameters; QMNAME is not required for the default queue manager, and the port number is not required if you are using the default (1414). The port number must not exceed 65535.

Note: To select which TCP/IP port number to use, IBM MQ uses the first port number it finds in the following sequence:

1. The port number explicitly specified in the channel definition or command line. This number allows the default port number to be overridden for a channel.
2. The port attribute specified in the TCP stanza of the `.ini` file. This number allows the default port number to be overridden for a queue manager.
3. The default value of 1414. This is the number assigned to IBM MQ by the Internet Assigned Numbers Authority for both inbound and outbound connections.

For the best performance, run the IBM MQ listener as a trusted application as described in “Running channels and listeners as trusted applications” on page 173. See Restrictions for trusted applications for information about trusted applications

Using the TCP/IP SO_KEEPALIVE option

If you want to use the Windows SO_KEEPALIVE option you must add the following entry to your registry:

```
TCP:
KeepAlive=yes
```

For more information about the SO_KEEPALIVE option, see “Checking that the other end of the channel is still available” on page 159.

On Windows, the `HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters` registry value for the Windows KeepAliveTime option controls the interval that elapses before the connection is checked. The default is two hours.

Using the TCP listener backlog option

In TCP, connections are treated incomplete unless three-way handshake takes place between the server and the client. These connections are called outstanding connection requests. A maximum value is set for these outstanding connection requests and can be considered a backlog of requests waiting on the TCP port for the listener to accept the request.

See “Using the TCP listener backlog option” on page 192 for more information, and the specific value for Windows.

Defining an LU 6.2 connection on Windows

SNA must be configured so that an LU 6.2 conversation can be established between the two machines.

Once the SNA is configured, proceed as follows.

See the following table for information.

Table 21. Settings on the local Windows system for a remote queue manager platform

Remote platform	TPNAME	TPPATH
z/OS or MVS™/ESA without CICS	The same as in the corresponding side information about the remote queue manager.	-
z/OS or MVS/ESA using CICS	CKRC (sender) CKSV (requester) CKRC (server)	-
IBM i	The same as the compare value in the routing entry on the IBM i system.	-
UNIX and Linux systems	The same as in the corresponding side information about the remote queue manager.	*MQ_INSTALLATION_PATH*/bin/amqcrs6a
Windows	As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.	*MQ_INSTALLATION_PATH*\bin\amqcrs6a

MQ_INSTALLATION_PATH represents the high-level directory in which IBM MQ is installed.

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique.

For the latest information about configuring AnyNet® SNA over TCP/IP, see the following online IBM documentation: AnyNet SNA over TCP/IP and SNA Node Operations.

Related concepts

“Sending end on LU 6.2” on page 186

Create a CPI-C side object (symbolic destination) from the administration application of the LU 6.2 product you are using. Enter this name in the Connection name field in the channel definition. Also create an LU 6.2 link to the partner.

"Receiving on LU 6.2" on page 186
Receiving channel programs are started in response to a startup request from the sending channel.

Sending end on LU 6.2
Create a CPI-C side object (symbolic destination) from the administration application of the LU 6.2 product you are using. Enter this name in the Connection name field in the channel definition. Also create an LU 6.2 link to the partner.

In the CPI-C side object enter the partner LU Name at the receiving machine, the TP Name and the Mode Name. For example:

```
Partner LU Name              OS2ROG2
Partner TP Name              recv
Mode Name                    #INTER
```

Receiving on LU 6.2
Receiving channel programs are started in response to a startup request from the sending channel.

To start a receiving channel program, a listener program has to be started to detect incoming network requests and start the associated channel. You start this listener program with the RUNMQLSR command, giving the TpName to listen on. Alternatively, you can use TpStart under SNA Server for Windows.

Using the RUNMQLSR command

Example of the command to start the listener:

```
RUNMQLSR -t LU62 -n RECV [-m QMNAME]
```

where RECV is the TpName that is specified at the other (sending) end as the "TpName to start on the remote side". The last part in square brackets is optional and is not required for the default queue manager.

It is possible to have more than one queue manager running on one machine. You must assign a different TpName to each queue manager, and then start a listener program for each one. For example:

```
RUNMQLSR -t LU62 -m QM1 -n TpName1
RUNMQLSR -t LU62 -m QM2 -n TpName2
```

For the best performance, run the IBM MQ listener as a trusted application as described in Running channels and listeners as trusted applications. See Restrictions for trusted applications for information about trusted applications.

You can stop all IBM MQ listeners running on a queue manager that is inactive, using the command:

```
ENDMQLSR [-m QMNAME]
```

Using Microsoft SNA Server on Windows

You can use TpSetup (from the SNA Server SDK) to define an invokable TP that then drives amqcrs6a.exe, or you can set various registry values manually. The parameters that should be passed to amqcrs6a.exe are:

```
-m QM -n TpName
```

where *QM* is the Queue Manager name and *TpName* is the TP Name. See the *Microsoft SNA Server APPC Programmers Guide* or the *Microsoft SNA Server CPI-C Programmers Guide* for more information.

If you do not specify a queue manager name, the default queue manager is assumed.

Defining a NetBIOS connection on Windows

IBM MQ uses three types of NetBIOS resource when establishing a NetBIOS connection to another IBM MQ product: sessions, commands, and names. Each of these resources has a limit, which is established either by default or by choice during the installation of NetBIOS.

Each running channel, regardless of type, uses one NetBIOS session and one NetBIOS command. The IBM NetBIOS implementation allows multiple processes to use the same local NetBIOS name. Therefore, only one NetBIOS name needs to be available for use by IBM MQ. Other vendors' implementations, for example Novell's NetBIOS emulation, require a different local name per process. Verify your requirements from the documentation for the NetBIOS product you are using.

In all cases, ensure that sufficient resources of each type are already available, or increase the maximums specified in the configuration. Any changes to the values require a system restart.

During system startup, the NetBIOS device driver displays the number of sessions, commands, and names available for use by applications. These resources are available to any NetBIOS-based application that is running on the same system. Therefore, it is possible for other applications to consume these resources before IBM MQ needs to acquire them. Your LAN network administrator should be able to clarify this for you.

Related concepts

"Defining the IBM MQ local NetBIOS name" on page 187
The local NetBIOS name used by IBM MQ channel processes can be specified in three ways.

"Establishing the queue manager NetBIOS session, command, and name limits" on page 188
The queue manager limits for NetBIOS sessions, commands, and names can be specified in two ways.

"Establishing the LAN adapter number" on page 188
For channels to work successfully across NetBIOS, the adapter support at each end must be compatible. IBM MQ allows you to control the choice of LAN adapter (LANA) number by using the AdapterNum value in the NETBIOS stanza of your qm.ini file and by specifying the `-a` parameter on the runmqlsr command.

"Initiating the NetBIOS connection" on page 188
Defining the steps needed to initiate a connection.

"Target listener for the NetBIOS connection" on page 189
Defining the steps to be undertaken at the receiving end of the NetBIOS connection.

Defining the IBM MQ local NetBIOS name

The local NetBIOS name used by IBM MQ channel processes can be specified in three ways.

In order of precedence the three ways are:

1. The value specified in the `-l` parameter of the RUNMQLSR command, for example:

   ```
   RUNMQLSR -t NETBIOS -l my_station
   ```

2. The MQNAME environment variable with a value that is established by the command:

   ```
   SET MQNAME= my_station
   ```

   You can set the MQNAME value for each process. Alternatively, you can set it at a system level in the Windows registry.

   If you are using a NetBIOS implementation that requires unique names, you must issue a SET MQNAME command in each window in which an IBM MQ process is started. The MQNAME value is arbitrary but it must be unique for each process.

3. The NETBIOS stanza in the queue manager configuration file qm.ini. For example:

```
NETBIOS:

LocalName= my_station
```

Note:

1. Due to the variations in implementation of the NetBIOS products supported, you are advised to make each NetBIOS name unique in the network. If you do not, unpredictable results might occur. If you have problems establishing a NetBIOS channel and there are error messages in the queue-manager error log showing a NetBIOS return code of X'15', review your use of NetBIOS names.
2. On Windows, you cannot use your machine name as the NetBIOS name because Windows already uses it.
3. Sender channel initiation requires that a NetBIOS name be specified either by using the MQNAME environment variable or the LocalName in the qm.ini file.

Establishing the queue manager NetBIOS session, command, and name limits
The queue manager limits for NetBIOS sessions, commands, and names can be specified in two ways.

In order of precedence these ways are:

1. The values specified in the RUNMQLSR command:

   ```
   -s Sessions
   -e Names
   -o Commands
   ```

   If the -m operand is not specified in the command, the values apply only to the default queue manager.
2. The NETBIOS stanza in the queue manager configuration file qm.ini. For example:

   ```
   NETBIOS:

   NumSess= Qmgr_max_sess
   NumCmds= Qmgr_max_cmds
   NumNames= Qmgr_max_names
   ```

Establishing the LAN adapter number
For channels to work successfully across NetBIOS, the adapter support at each end must be compatible. IBM MQ allows you to control the choice of LAN adapter (LANA) number by using the AdapterNum value in the NETBIOS stanza of your qm.ini file and by specifying the `-a` parameter on the runmqlsr command.

The default LAN adapter number used by IBM MQ for NetBIOS connections is 0. Verify the number being used on your system as follows:

On Windows, it is not possible to query the LAN adapter number directly through the operating system. Instead, you use the LANACFG.EXE command-line utility, available from Microsoft. The output of the tool shows the virtual LAN adapter numbers and their effective bindings. For further information about LAN adapter numbers, see the Microsoft Knowledge Base article 138037 *HOWTO: Use LANA Numbers in a 32-bit Environment*.

Specify the correct value in the NETBIOS stanza of the queue manager configuration file, qm.ini:

```
NETBIOS:
AdapterNum= n
```

where n is the correct LAN adapter number for this system.

Initiating the NetBIOS connection
Defining the steps needed to initiate a connection.

To initiate the connection, follow these steps at the sending end:

1. Define the NetBIOS station name using the MQNAME or LocalName value.
2. Verify the LAN adapter number being used on your system and specify the correct file using the AdapterNum.
3. In the ConnectionName field of the channel definition, specify the NetBIOS name being used by the target listener program. On Windows, NetBIOS channels ***must*** be run as threads. Do this by specifying MCATYPE(THREAD) in the channel definition.

   ```
   DEFINE CHANNEL (chname) CHLTYPE(SDR) +
   TRPTYPE(NETBIOS) +
   CONNAME(your_station) +
   XMITQ(xmitq) +
   MCATYPE(THREAD) +
   REPLACE
   ```

Target listener for the NetBIOS connection
Defining the steps to be undertaken at the receiving end of the NetBIOS connection.

At the receiving end, follow these steps:

1. Define the NetBIOS station name using the MQNAME or LocalName value.
2. Verify the LAN adapter number being used on your system and specify the correct file using the AdapterNum.
3. Define the receiver channel:

   ```
   DEFINE CHANNEL (chname) CHLTYPE(RCVR) +
   TRPTYPE(NETBIOS) +
   REPLACE
   ```

4. Start the IBM MQ listener program to establish the station and make it possible to contact it. For example:

   ```
   RUNMQLSR -t NETBIOS -l your_station [-m qmgr]
   ```

   This command establishes `your_station` as a NetBIOS station waiting to be contacted. The NetBIOS station name must be unique throughout your NetBIOS network.

For the best performance, run the IBM MQ listener as a trusted application as described in “Running channels and listeners as trusted applications” on page 173. See Restrictions for trusted applications for information about trusted applications.

You can stop all IBM MQ listeners running on a queue manager that is inactive, using the command:

```
ENDMQLSR [-m QMNAME]
```

If you do not specify a queue manager name, the default queue manager is assumed.

Setting up communication on UNIX and Linux systems

DQM is a remote queuing facility for IBM MQ. It provides channel control programs for the queue manager which form the interface to communication links, controllable by the system operator. The channel definitions held by distributed-queuing management use these connections.

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. To succeed, it is necessary for the connection to be defined and available. This section explains how to do this. You might also find it helpful to refer to the following sections:

- Example configuration - IBM MQ for AIX
- Example configuration - IBM MQ for HP-UX
- Example configuration - IBM MQ for Solaris

- Example configuration - IBM MQ for Linux

For Windows, see “Setting up communication for Windows” on page 183.

You can choose between two forms of communication for IBM MQ on UNIX and Linux systems:

- “Defining a TCP connection on UNIX and Linux” on page 190
- “Defining an LU 6.2 connection on UNIX and Linux” on page 193

Each channel definition must specify one only as the transmission protocol (Transport Type) attribute. One or more protocols can be used by a queue manager.

For IBM MQ MQI clients, it might be useful to have alternative channels using different transmission protocols. For more information about IBM MQ MQI clients, see Overview of IBM MQ MQI clients.

Related concepts

“Configuring distributed queuing” on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

“Monitoring and controlling channels on Windows, UNIX and Linux platforms” on page 173
For DQM you need to create, monitor, and control the channels to remote queue managers. You can control channels using commands, programs, IBM MQ Explorer, files for the channel definitions, and a storage area for synchronization information.

“Configuring connections between the client and server” on page 14
To configure the communication links between IBM MQ MQI clients and servers, decide on your communication protocol, define the connections at both ends of the link, start a listener, and define channels.

Defining a TCP connection on UNIX and Linux

The channel definition at the sending end specifies the address of the target. The listener or inet daemon is configured for the connection at the receiving end.

Sending end

Specify the host name, or the TCP address of the target machine, in the Connection Name field of the channel definition. The port to connect to defaults to 1414. Port number 1414 is assigned by the Internet Assigned Numbers Authority to IBM MQ.

To use a port number other than the default, change the connection name field thus:

```
Connection Name REMHOST(1822)
```

where REMHOST is the host name of the remote machine and 1822 is the port number required. (This must be the port that the listener at the receiving end is listening on.)

Alternatively you can change the port number by specifying it in the queue manager configuration file (qm.ini):

```
TCP:
Port=1822
```

For more information about the values you set using qm.ini, see Configuration file stanzas for distributed queuing.

Receiving on TCP

You can use either the TCP/IP listener, which is the inet daemon (inetd), or the IBM MQ listener.

Some Linux distributions now use the extended inet daemon (xinetd) instead of the inet daemon. For information about how to use the extended inet daemon on a Linux system, see Establishing a TCP connection on Linux .

Related concepts

"Using the TCP/IP listener" on page 191
To start channels on UNIX and Linux, the /etc/services file and the inetd.conf file must be edited

"Using the TCP listener backlog option" on page 192
In TCP, connections are treated incomplete unless three-way handshake takes place between the server and the client. These connections are called outstanding connection requests. A maximum value is set for these outstanding connection requests and can be considered a backlog of requests waiting on the TCP port for the listener to accept the request.

"Using the IBM MQ listener" on page 193
To run the listener supplied with IBM MQ, which starts new channels as threads, use the `runmqlsr` command.

"Using the TCP/IP SO_KEEPALIVE option" on page 193
On some UNIX and Linux systems, you can define how long TCP waits before checking that the connection is still available, and how frequently it tries the connection again if the first check fails. This is either a kernel tunable parameter, or can be entered at the command line.

Using the TCP/IP listener
To start channels on UNIX and Linux, the /etc/services file and the inetd.conf file must be edited

Follow these instructions:

1. Edit the /etc/services file:

   Note: To edit the /etc/services file, you must be logged in as a superuser or root. You can change this, but it must match the port number specified at the sending end.

   Add the following line to the file:

   ```
   MQSeries 1414/tcp
   ```

   where 1414 is the port number required by IBM MQ. The port number must not exceed 65535.
2. Add a line in the inetd.conf file to call the program amqcrsta, where *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed:

   ```
   MQSeries stream tcp nowait mqm MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta
   [-m Queue_Man_Name]
   ```

The updates are active after inetd has reread the configuration files. To do this, issue the following commands from the root user ID:

- On AIX:

  ```
  refresh -s inetd
  ```
- On HP-UX, from the mqm user ID:

  ```
  inetd -c
  ```
- On Solaris 10 or later:

  ```
  inetconv
  ```
- On other UNIX and Linux systems (including Solaris 9):

  ```
  kill -1 < process number >
  ```

When the listener program started by inetd inherits the locale from inetd, it is possible that the MQMDE is not honored (merged) and is placed on the queue as message data. To ensure that the MQMDE is honored, you must set the locale correctly. The locale set by inetd might not match that chosen for other locales used by IBM MQ processes. To set the locale:

1. Create a shell script which sets the locale environment variables LANG, LC_COLLATE, LC_CTYPE, LC_MONETARY, LC_NUMERIC, LC_TIME, and LC_MESSAGES to the locale used for other IBM MQ process.
2. In the same shell script, call the listener program.
3. Modify the inetd.conf file to call your shell script in place of the listener program.

It is possible to have more than one queue manager on the server. You must add a line to each of the two files, for each of the queue managers. For example:

```
MQSeries1   1414/tcp
MQSeries2   1822/tcp
```

```
MQSeries2 stream tcp nowait mqm MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta -m QM2
```

Where *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.

This avoids error messages being generated if there is a limitation on the number of outstanding connection requests queued at a single TCP port. For information about the number of outstanding connection requests, see "Using the TCP listener backlog option" on page 192.

Using the TCP listener backlog option
In TCP, connections are treated incomplete unless three-way handshake takes place between the server and the client. These connections are called outstanding connection requests. A maximum value is set for these outstanding connection requests and can be considered a backlog of requests waiting on the TCP port for the listener to accept the request.

The default listener backlog values are shown in Table 22 on page 192.

Table 22. Maximum outstanding connection requests queued at a TCP/IP port

Server platform	**Maximum connection requests**
AIX	100
HP-UX	20
Linux	100
IBM i	255
Solaris	100
Windows Server	100
Windows Workstation	100
z/OS	255

If the backlog reaches the values shown in Table 22 on page 192, the TCP/IP connection is rejected and the channel is not able to start.

For MCA channels, this results in the channel going into a RETRY state and trying the connection again at a later time.

However, to avoid this error, you can add an entry in the qm.ini file:

```
TCP:
ListenerBacklog = n
```

This overrides the default maximum number of outstanding requests (see Table 22 on page 192) for the TCP/IP listener.

Note: Some operating systems support a larger value than the default. If necessary, this value can be used to avoid reaching the connection limit.

To run the listener with the `backlog` option switched on either:

- Use the `runmqlsr -b` command, or
- Use the MQSC command **`DEFINE LISTENER`** with the BACKLOG attribute set to the required value.

For information about the **`runmqlsr`** command, see runmqlsr. For information about the DEFINE LISTENER command, see DEFINE LISTENER.

Using the IBM MQ listener
To run the listener supplied with IBM MQ, which starts new channels as threads, use the `runmqlsr` command.

For example:

```
runmqlsr -t tcp [-m QMNAME] [-p 1822]
```

The square brackets indicate optional parameters; QMNAME is not required for the default queue manager, and the port number is not required if you are using the default (1414). The port number must not exceed 65535.

For the best performance, run the IBM MQ listener as a trusted application as described in “Running channels and listeners as trusted applications” on page 173. See Restrictions for trusted applications for information about trusted applications.

You can stop all IBM MQ listeners running on a queue manager that is inactive, using the command:

```
endmqlsr [-m QMNAME]
```

If you do not specify a queue manager name, the default queue manager is assumed.

Using the TCP/IP SO_KEEPALIVE option
On some UNIX and Linux systems, you can define how long TCP waits before checking that the connection is still available, and how frequently it tries the connection again if the first check fails. This is either a kernel tunable parameter, or can be entered at the command line.

If you want to use the SO_KEEPALIVE option (for more information, see “Checking that the other end of the channel is still available” on page 159) you must add the following entry to your queue manager configuration file (qm.ini):

```
TCP:
KeepAlive=yes
```

See the documentation for your UNIX and Linux system for more information.

Defining an LU 6.2 connection on UNIX and Linux

SNA must be configured so that an LU 6.2 conversation can be established between the two machines.

For the latest information about configuring SNA over TCP/IP, see the following online IBM documentation: Communications Server.

SNA must be configured so that an LU 6.2 conversation can be established between the two systems.

See the *Multiplatform APPC Configuration Guide* and the following table for information.

Table 23. Settings on the local UNIX and Linux system for a remote queue manager platform		
Remote platform	**TPNAME**	**TPPATH**
z/OS without CICS	The same as the corresponding TPName in the side information about the remote queue manager.	-
z/OS using CICS	CKRC (sender) CKSV (requester) CKRC (server)	-
IBM i	The same as the compare value in the routing entry on the IBM i system.	-
UNIX and Linux systems	The same as the corresponding TPName in the side information about the remote queue manager.	*MQ_INSTALLATION_PATH*/bin/amqcrs6a
Windows	As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.	*MQ_INSTALLATION_PATH*\bin\amqcrs6a

MQ_INSTALLATION_PATH represents the high-level directory in which IBM MQ is installed.

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique.

Related concepts

"Sending end" on page 194
On UNIX and Linux systems, create a CPI-C side object (symbolic destination) and enter this name in the Connection name field in the channel definition. Also create an LU 6.2 link to the partner.

"Receiving on LU 6.2" on page 194
On UNIX and Linux systems, create a listening attachment at the receiving end, an LU 6.2 logical connection profile, and a TPN profile.

Sending end

On UNIX and Linux systems, create a CPI-C side object (symbolic destination) and enter this name in the Connection name field in the channel definition. Also create an LU 6.2 link to the partner.

In the CPI-C side object enter the partner LU name at the receiving machine, the transaction program name and the mode name. For example:

```
Partner LU Name              REMHOST
Remote TP Name               recv
Service Transaction Program  no
Mode Name                    #INTER
```

On HP-UX, use the APPCLLU environment variable to name the local LU that the sender should use. On Solaris, set the APPC_LOCAL_LU environment variable to be the local LU name.

SECURITY PROGRAM is used, where supported by CPI-C, when IBM MQ attempts to establish an SNA session.

Receiving on LU 6.2

On UNIX and Linux systems, create a listening attachment at the receiving end, an LU 6.2 logical connection profile, and a TPN profile.

In the TPN profile, enter the full path to the executable file and the Transaction Program name:

```
Full path to TPN executable    MQ_INSTALLATION_PATH/bin/amqcrs6a
```

```
Transaction Program name      recv
User ID                       0
```

MQ_INSTALLATION_PATH represents the high-level directory in which IBM MQ is installed.

On systems where you can set the user ID, specify a user who is a member of the mqm group. On AIX, Solaris, and HP-UX, set the APPCTPN (transaction name) and APPCLLU (local LU name) environment variables (you can use the configuration panels for the invoked transaction program).

You might need to use a queue manager other than the default queue manager. If so, define a command file that calls:

```
amqcrs6a -m Queue_Man_Name
```

then call the command file.

Monitoring and controlling channels on IBM i

Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers. Each queue manager has a DQM program for controlling interconnections to compatible remote queue managers.

The following list is a brief description of the components of the channel control function:

- Channel definitions are held as queue manager objects.
- The channel commands are a subset of the IBM MQ for IBM i set of commands.

  Use the command GO CMDMQM to display the full set of IBM MQ for IBM i commands.
- You use channel definition panels, or commands to:
  - Create, copy, display, change, and delete channel definitions
  - Start and stop channels, ping, reset channel sequence numbers, and resolve in-doubt messages when links cannot be re-established
  - Display status information about channels
- Channels can also be managed using MQSC
- Channels can also be managed using IBM MQ Explorer
- Sequence numbers and *logical unit of work (LUW)* identifiers are stored in the synchronization file, and are used for channel synchronization purposes.

You can use the commands and panels to: define message channels and associated objects, and monitor and control message channels. By using the F4=Prompt key, you can specify the relevant queue manager. If you do not use the prompt, the default queue manager is assumed. With F4=Prompt, an additional panel is displayed where you can enter the relevant queue manager name and sometimes other data.

The objects you need to define with the panels are:

- Transmission queues
- Remote queue definitions
- Queue manager alias definitions
- Reply-to queue alias definitions
- Reply-to local queues
- Message channel definitions

For more information about the concepts involved in the use of these objects, see "Configuring distributed queuing" on page 122.

Channels must be completely defined, and their associated objects must exist and be available for use, before a channel can be started.

In addition, the particular communication link for each channel must be defined and available before a channel can be run. For a description of how LU 6.2 and TCP/IP links are defined, see the particular communication guide for your installation.

For more information about creating and working with objects, see:

- "Creating objects" on page 196
- "Creating a channel" on page 196
- "Starting a channel" on page 198
- "Selecting a channel" on page 199
- "Browsing a channel" on page 199
- "Renaming a channel" on page 201
- "Work with channel status" on page 201
- "Work-with-channel choices" on page 202

Related concepts

"Setting up communication for IBM MQ for IBM i" on page 208
When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For it to succeed, it is necessary for the connection to be defined and available.

"Configuring connections between the client and server" on page 14
To configure the communication links between IBM MQ MQI clients and servers, decide on your communication protocol, define the connections at both ends of the link, start a listener, and define channels.

Related reference

Example configuration - IBM MQ for IBM i
Message channel planning example for IBM MQ for IBM i
IBM MQ for IBM i CL commands

IBM i Creating objects

You can use the CRTMQMQ command to create the queue and alias objects.

You can create the queue and alias objects, such as: transmission queues, remote queue definitions, queue manager alias definitions, reply-to queue alias definitions, and reply-to local queues.

For a list of default objects, see IBM MQ for IBM i system and default objects.

Creating a channel

You can create a channel from the Create Channel panel or by using the CRTMQMCHL command on the command line.

To create a channel:

1. Use F6 from the Work with MQM Channels panel (WRKMQMCHL).

   Alternatively, use the CRTMQMCHL command from the command line.

   Either way, the Create Channel panel is displayed. Type:

   - The name of the channel in the field provided
   - The channel type for this end of the link
2. Press enter.

Note: You must name all the channels in your network uniquely. As shown in Network diagram showing all channels, including the source and target queue manager names in the channel name is a good way to do so.

Your entries are validated and errors are reported immediately. Correct any errors and continue.

You are presented with the appropriate channel settings panel for the type of channel you have chosen. Complete the fields with the information you have gathered previously. Press enter to create the channel.

You are provided with help in deciding on the content of the various fields in the descriptions of the channel definition panels in the help panels, and in Channel attributes.

```
Create MQM Channel (CRTMQMCHL)

Type choices, press Enter.

Channel name . . . . . . . . . . > CHANNAME________________
Channel type . . . . . . . . . . > *SDR__    *RCVR, *SDR, *SVR, *RQSTR...
Message Queue Manager name     *DFT___________________________________
_____
Replace . . . . . . . . . . . .  *NO_     *NO, *YES
Transport type . . . . . . . . .  *TCP____   *LU62, *TCP, *SYSDFTCHL
Text 'description' . . . . . . . > 'Example Channel Definition'______________
___________________________________
Connection name . . . . . . . .  *SYSDFTCHL_________________________________
______________________________________________________________________________
______________________________________________________________________________
______________________________________________________________________________
______________________________________________________________________________
______________________________________________________________________________
__________________________________________________
More...
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 25. Create channel (1)

```
Create MQM Channel (CRTMQMCHL)

Type choices, press Enter.

Transmission queue . . . . . . .  'TRANSMISSION_QUEUE_NAME'__________________
__________________
Message channel agent . . . . .  *NONE______  Name, *SYSDFTCHL, *NONE
Library . . . . . . . . . . .  __________  Name
Message channel agent user ID .  *SYSDFTCHL__ Character value...
Coded Character Set Identifier   *SYSDFTCHL__ 0-9999, *SYSDFTCHL
Batch size . . . . . . . . . . .  50_________  1-9999, *SYSDFTCHL
Disconnect interval . . . . . .  6000_______  1-999999, *SYSDFTCHL
Short retry interval . . . . . .  60_________  0-999999999, *SYSDFTCHL
Short retry count . . . . . . .  10_________  0-999999999, *SYSDFTCHL
Long retry interval . . . . . .  1200_______  0-999999999, *SYSDFTCHL
Long retry count . . . . . . . .  999999999__  0-999999999, *SYSDFTCHL
Security exit . . . . . . . . .  *NONE_____  Name, *SYSDFTCHL, *NONE
Library . . . . . . . . . . .  __________  Name
Security exit user data . . . .  *SYSDFTCHL______________________

More...
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 26. Create channel (2)

```
Create MQM Channel (CRTMQMCHL)

Type choices, press Enter.

Send exit . . . . . . . . . . .  *NONE______  Name, *SYSDFTCHL, *NONE
Library . . . . . . . . . . .  ___________ Name
+ for more values   __________
Send exit user data . . . . . .  ________________________________
+ for more values  ________________________________
Receive exit . . . . . . . . . .  *NONE_____  Name, *SYSDFTCHL, *NONE
Library . . . . . . . . . . .  __________  Name
+ for more values  __________
__________
Receive exit user data . . . . .  ________________________________
+ for more values  ________________________________
Message exit . . . . . . . . . .  *NONE_____  Name, *SYSDFTCHL, *NONE
Library . . . . . . . . . . .  __________  Name
+ for more values  __________
__________
More...
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 27. Create channel (3)

```
Create MQM Channel (CRTMQMCHL)

Type choices, press Enter.

Message exit user data . . . . .  ________________________________
+ for more values  ______________________________
Convert message . . . . . . . . .  *SYSDFTCHL_  *YES, *NO, *SYSDFTCHL
Sequence number wrap . . . . . .  999999999__  100-999999999, *SYSDFTCHL
Maximum message length . . . . .  4194304____  0-4194304, *SYSDFTCHL
Heartbeat interval . . . . . . .  300________  0-999999999, *SYSDFTCHL
Non Persistent Message Speed . .  *FAST_____  *FAST, *NORMAL, *SYSDFTCHL
Password . . . . . . . . . . . .  *SYSDFTCHL_  Character value, *BLANK...
Task User Profile . . . . . . .  *SYSDFTCHL_  Character value, *BLANK...
Transaction Program Name . . . .  *SYSDFTCHL




Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 28. Create channel (4)

Starting a channel

You can start a channel from the Work with Channels panel or by using the STRMQMCHL command on the command line.

Listeners are valid for TCP only. For SNA listeners, you must configure your communications subsystem.

For applications to be able to exchange messages, you must start a listener program for inbound connections using the STRMQMLSR command.

For outbound connections, you must start the channel in one of the following ways:

1. Use the CL command STRMQMCHL, specifying the channel name, to start the channel as a process or a thread, depending on the MCATYPE parameter. (If channels are started as threads, they are threads of a channel initiator.)

```
STRMQMCHL CHLNAME(QM1.TO.QM2) MQNAME(MYQMGR)
```

2. Use a channel initiator to trigger the channel. One channel initiator is started automatically when the queue manager is started. This automatic start can be eliminated by changing the chinit stanza in the qm.ini file for that queue manager.
3. Use the WRKMQMCHL command to begin the Work with Channels panel and choose option 14 to start a channel.

Selecting a channel

You can select a channel from the Work With channels panel.

To select a channel, use the WRKMQMCHL command to begin at the Work with Channels panel:

1. Move the cursor to the option field associated with the required channel name.
2. Type an option number.
3. Press enter to activate your choice.

If you select more than one channel, the options are activated in sequence.

```
Work with MQM Channels

Queue Manager Name . . :  CNX

Type options, press Enter.
2=Change  3=Copy  4=Delete  5=Display  8=Work with Status  13=Ping
14=Start  15=End  16=Reset  17=Resolve

Opt   Name           Type    Transport   Status
CHLNIC          *RCVR    *TCP      INACTIVE
CORSAIR.TO.MUSTANG    *SDR    *LU62    INACTIVE
FV.CHANNEL.MC.DJE1    *RCVR    *TCP     INACTIVE
FV.CHANNEL.MC.DJE2    *SDR    *TCP     INACTIVE
FV.CHANNEL.MC.DJE3    *RQSTR   *TCP     INACTIVE
FV.CHANNEL.MC.DJE4    *SVR    *TCP     INACTIVE
FV.CHANNEL.PETER     *RCVR    *TCP     INACTIVE
FV.CHANNEL.PETER.LU   *RCVR    *LU62    INACTIVE
FV.CHANNEL.PETER.LU1   *RCVR    *LU62    INACTIVE
More...
Parameters or command
===>
F3=Exit  F4=Prompt  F5=Refresh  F6=Create  F9=Retrieve  F12=Cancel
F21=Print
```

Figure 29. Work with channels

Browsing a channel

You can browse a channel from the Display Channel panel or by using the DSPMQMCHL command on the command line.

To browse the settings of a channel, use the WRKMQMCHL command to begin at the Display Channel panel:

1. Type option 5 (Display) against the required channel name.
2. Press enter to activate your choice.

If you select more than one channel, they are presented in sequence.

Alternatively, you can use the DSPMQMCHL command from the command line.

This results in the appropriate Display Channel panel being displayed with details of the current settings for the channel. The fields are described in Channel attributes.

```
Display MQM Channel

Channel name . . . . . . . . . :  ST.JST.2TO1
Queue Manager Name . . . . . . :  QMREL
Channel type . . . . . . . . . :  *SDR
Transport type . . . . . . . . :  *TCP
Text 'description' . . . . . . :  John's sender to WINSDOA1

Connection name . . . . . . . :  MUSTANG


Transmission queue . . . . . . :  WINSDOA1

Message channel agent . . . . :
Library . . . . . . . . . . :
Message channel agent user ID :  *NONE
Batch interval . . . . . . . . :  0
Batch size . . . . . . . . . . :  50
Disconnect interval . . . . . :  6000



F3=Exit  F12=Cancel  F21=Print
```

Figure 30. Display a TCP/IP channel (1)

```
Display MQM Channel

Short retry interval . . . . . :  60
Short retry count . . . . . . :  10
Long retry interval . . . . . :  6000
Long retry count . . . . . . . :  10
Security exit . . . . . . . . :
Library . . . . . . . . . . :
Security exit user data . . . :
Send exit . . . . . . . . . . :
Library . . . . . . . . . . :
Send exit user data . . . . . :
Receive exit . . . . . . . . . :
Library . . . . . . . . . . :
Receive exit user data . . . . :
Message exit . . . . . . . . . :
Library . . . . . . . . . . :
Message exit user data . . . . :
More...



F3=Exit  F12=Cancel  F21=Print
```

Figure 31. Display a TCP/IP channel (2)

```
Display MQM Channel

Sequence number wrap . . . . . :  999999999
Maximum message length . . . . :  10000
Convert message . . . . . . . :  *NO
Heartbeat interval . . . . . . .  300
Nonpersistent message speed . .  *FAST









Bottom





F3=Exit  F12=Cancel  F21=Print
```

Figure 32. Display a TCP/IP channel (3)

Renaming a channel

You can rename a channel from the Work with Channels panel.

To rename a message channel, begin at the Work with Channels panel:

1. End the channel.
2. Use option 3 (Copy) to create a duplicate with the new name.
3. Use option 5 (Display) to check that it has been created correctly.
4. Use option 4 (Delete) to delete the original channel.

If you decide to rename a message channel, ensure that both channel ends are renamed at the same time.

Work with channel status

You can work with the channel status from the Work with Channel Status panel.

Use the WRKMQMCHST command to display the first of a set of panels showing the status of your channels. You can view the status panels in sequence when you select Change-view (F11).

Alternatively, selecting option 8 (Work with Status) from the Work with MQM Channels panel also displays the first status panel.

```
MQSeries Work with Channel Status

Type options, press Enter.
5=Display  13=Ping  14=Start  15=End  16=Reset  17=Resolve


Opt Name         Connection      Indoubt   Last Seq
CARTS_CORSAIR_CHAN  GBIBMIYA.WINSDOA1    NO        1
CHLNIC        9.20.2.213       NO       3
FV.CHANNEL.PETER2   9.20.2.213       NO       6225
JST.1.2        9.20.2.201       NO       28
MP_MUST_TO_CORS    9.20.2.213       NO       100
MUSTANG.TO.CORSAIR  GBIBMIYA.WINSDOA1    NO        10
MP_CORS_TO_MUST    9.20.2.213       NO       101
JST.2.3        9.5.7.126        NO       32
PF_WINSDOA1_LU62   GBIBMIYA.IYA80020    NO       54
PF_WINSDOA1_LU62   GBIBMIYA.WINSDOA1    NO       500
ST.JCW.EXIT.2T01.CHL 9.20.2.213       NO       216

Bottom
Parameters or command
===>
F3=Exit  F4=Prompt  F5=Refresh  F6=Create  F9=Retrieve  F11=Change view
F12=Cancel  F21=Print
```

Figure 33. First of the set of channel status panels

The options available in the Work with Channel Status panel are:

Menu option	Description
5=Display	Displays the channel settings.
13=Ping	Initiates a Ping action, where appropriate.
14=Start	Starts the channel.
15=End	Stops the channel.
16=Reset	Resets the channel sequence number.
17=Resolve	Resolves an in-doubt channel situation, manually.

Work-with-channel choices

The Work with Channels panel is reached with the command WRKMQMCHL, and it allows you to monitor the status of all channels listed, and to issue commands against selected channels.

The options available in the Work with Channel panel are:

Menu option	Description
"2=Change" on page 203	Changes the attributes of a channel.
"3=Copy" on page 203	Copies the attributes of a channel to a new channel.
"4=Delete" on page 203	Deletes a channel.
"5=Display" on page 203	Displays the current settings for the channel.
"6=Create" on page 203	Displays the Create channel panel
"8=Work with Status" on page 204	Displays the channel status panels.
"13=Ping" on page 205	Runs the Ping facility to test the connection to the adjacent system by exchanging a fixed data message with the remote end.
"14=Start" on page 205	Starts the selected channel, or resets a disabled receiver channel.

Menu option	Description
“15=End” on page 206	Requests the channel to close down.
“16=Reset” on page 207	Requests the channel to reset the sequence numbers on this end of the link. The numbers must be equal at both ends for the channel to start.
“17=Resolve” on page 207	Requests the channel to resolve in-doubt messages without establishing connection to the other end.
“18=Display authority” on page 207	Displays IBM MQ object authority
“19=Grant authority” on page 207	Grants IBM MQ object authority
“20=Revoke authority” on page 207	Revokes IBM MQ object authority
“21=Recover object” on page 207	Recovers IBM MQ object
“22=Record image” on page 208	Records IBM MQ object image

2=Change

Use the Change option to change an existing channel definition.

The Change option, or the CHGMQMCHL command, changes an existing channel definition, except for the channel name. Type over the fields to be changed in the channel definition panel, and then save the updated definition by pressing enter.

3=Copy

Use the Copy option to copy an existing channel.

The Copy option uses the CPYMQMCHL command to copy an existing channel. The Copy panel enables you to define the new channel name. However, you must restrict the characters used to those characters that are valid for IBM MQ for IBM i object names; see the *IBM MQ for IBM i System Administration*.

Press enter on the Copy panel to display the details of current settings. You can change any of the new channel settings. Save the new channel definition by pressing enter.

4=Delete

Use the Delete option to delete the selected channel.

A panel is displayed to confirm or cancel your request.

5=Display

Use the Display option to display the current definitions for the channel.

This choice displays the panel with the fields showing the current values of the parameters, and protected against user input.

6=Create

Use the Create option to display the Create channel panel.

Use the Create option, or enter the CRTMQMCHL command from the command line, to obtain the Create Channel panel. There are examples of Create Channel panels, starting at Figure 25 on page 197.

With this panel, you create a channel definition from a screen of fields filled with default values supplied by IBM MQ for IBM i. Type the name of the channel, select the type of channel you are creating, and the communication method to be used.

When you press enter, the panel is displayed. Type information in all the required fields in this panel, and the remaining panels, and then save the definition by pressing enter.

The channel name must be the same at both ends of the channel, and unique within the network. However, you must restrict the characters used to those characters that are valid for IBM MQ for IBM i object names.

All panels have default values supplied by IBM MQ for IBM i for some fields. You can customize these values, or you can change them when you are creating or copying channels. To customize the values, see the *IBM MQ for IBM i System Administration*.

You can create your own set of channel default values by setting up dummy channels with the required defaults for each channel type, and copying them each time you want to create new channel definitions.

Related reference

Channel attributes

8=Work with Status

Use Work with Status to see detailed channel status information.

The status column tells you whether the channel is active or inactive, and is displayed continuously in the Work with MQM Channels panel. Use option 8 (Work with Status) to see more status information displayed. Alternatively, this information can be displayed from the command line with the WRKMQMCHST command. See "Work with channel status" on page 201.

- Channel name
- Channel type
- Channel status
- Channel instance
- Remote queue manager
- Transmission queue name
- Communication connection name
- In-doubt status of channel
- Last sequence number
- Number of indoubt messages
- In-doubt sequence number
- Number of messages on transmission queue
- Logical unit of work identifier
- In-doubt logical unit of work identifier
- Channel substate
- Channel monitoring
- Header compression
- Message compression
- Compression time indicator
- Compression rate indicator
- Transmission queue time indicator
- Network time indicator
- Exit time indicator
- Batch size indicator
- Current shared conversations
- Maximum shared conversations

13=Ping

Use the Ping option to exchange a fixed data message with the remote end.

A successful IBM MQ Ping gives some confidence to the system supervisor that the channel is available and functioning.

Ping does not involve the use of transmission queues and target queues. It uses channel definitions, the related communication link, and the network setup.

It is available from sender and server channels, only. The corresponding channel is started at the far side of the link, and performs the start-up parameter negotiation. Errors are notified normally.

The result of the message exchange is presented in the Ping panel for you, and is the returned message text, together with the time the message was sent, and the time the reply was received.

Ping with LU 6.2

When Ping is invoked in IBM MQ for IBM i, it is run with the user ID of the user requesting the function, whereas the normal way that a channel program is run is for the QMQM user ID to be taken for channel programs. The user ID flows to the receiving side and it must be valid on the receiving end for the LU 6.2 conversation to be allocated.

14=Start

Use the Start option to start a channel manually.

The Start option is available for sender, server, and requester channels. It is not necessary where a channel has been set up with queue manager triggering.

The Start option is also used for receiver, server-connection, cluster sender, and cluster receiver channels. Starting a receiver channel that is in STOPPED state means that it can be started from the remote channel.

When started, the sending MCA reads the channel definition file and opens the transmission queue. A channel start-up sequence is issued, which remotely starts the corresponding MCA of the receiver or server channel. When they have been started, the sender and server processes await messages arriving on the transmission queue and transmit them as they arrive.

When you use triggering, you must start the continuously running trigger process to monitor the initiation queue. The STRMQMCHLI command can be used for starting the process.

At the far end of a channel, the receiving process might be started in response to a channel startup from the sending end. The method of doing so is different for LU 6.2 and TCP/IP connected channels:

- LU 6.2 connected channels do not require any explicit action at the receiving end of a channel.
- TCP connected channels require a listener process to be running continuously. This process awaits channel startup requests from the remote end of the link and starts the process defined in the channel definitions for that connection.

  When the remote system is IBM i, you can use the STRMQMLSR command.

Use of the Start option always causes the channel to resynchronize, where necessary.

For the start to succeed:

- Channel definitions, local and remote must exist. If there is no appropriate channel definition for a receiver or server-connection channel, a default one is created automatically if the channel is auto-defined. See Channel auto-definition exit program.
- The transmission queue must exist, be enabled for GETs, and have no other channels using it.
- MCAs, local and remote, must exist.
- The communication link must be available.
- The queue managers must be running, local and remote.
- The message channel must be inactive.

To transfer messages, remote queues and remote queue definitions must exist.

A message is returned to the panel confirming that the request to start a channel has been accepted. For confirmation that the Start process has succeeded, check the system log, or press F5 (refresh the screen).

15=End

Use End to stop channel activity

Use the End option to request the channel to stop activity. The channel does not send any more messages.

Select F4 before pressing enter to choose whether the channel becomes STOPPED or INACTIVE, and whether to stop the channel using a CONTROLLED or an IMMEDIATE stop. A stopped channel must be restarted by the operator to become active again. An inactive channel can be triggered.

Stop immediate

Use Stop immediate to stop a channel without completing any unit of work.

This option terminates the channel process. As a result the channel does not complete processing the current batch of messages, and cannot, therefore, leave the channel in doubt. In general, it is better for the operators to use the controlled stop option.

Stop controlled

Use Stop controlled to stop a channel at the end of the current unit of work.

This choice requests the channel to close down in an orderly way; the current batch of messages is completed, and the sync point procedure is carried out with the other end of the channel.

Restarting stopped channels

When a channel goes into STOPPED state, you must restart the channel manually. You can restart the channel in the following ways:

- By using the **START CHANNEL** MQSC command.
- By using the **Start Channel** PCF command.
- By using the MQ Explorer.
- z/OS On z/OS, by using the Start a channel panel.
- IBM i On IBM i, by using the **STRMQMCHL CL** command or the **START** option on the WRKMQMCHL panel.

For sender or server channels, when the channel entered the STOPPED state, the associated transmission queue was set to GET(DISABLED) and triggering was set off. When the start request is received, these attributes are reset automatically.

z/OS If the channel initiator stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the channel initiator is restarted. However, the channel status for the SVRCONN channel type is reset if the channel initiator stops while the channel is in STOPPED status.

IBM i distributed If the queue manager stops while a channel is in RETRYING or STOPPED status, the channel status is remembered when the queue manager is restarted. From IBM MQ Version 8.0 onwards, this applies to SVRCONN channels as well. Previously, the channel status for the SVRCONN channel type was reset if the channel initiator stopped while the channel was in STOPPED status.

16=Reset

Use the Reset option to force a new message sequence.

The Reset option changes the message sequence number. Use it with care, and only after you have used the Resolve option to resolve any in-doubt situations. This option is available only at the sender or server channel. The first message starts the new sequence the next time the channel is started.

17=Resolve

Use the Resolve option to force a local commit or backout of in-doubt messages held in a transmission queue.

Use the Resolve option when messages are held in-doubt by a sender or server, for example because one end of the link has terminated, and there is no prospect of it recovering. The Resolve option accepts one of two parameters: BACKOUT or COMMIT. Backout restores messages to the transmission queue, while Commit discards them.

The channel program does not try to establish a session with a partner. Instead, it determines the logical unit of work identifier (LUWID) which represents the in-doubt messages. It then issues, as requested, either:

- BACKOUT to restore the messages to the transmission queue; or
- COMMIT to delete the messages from the transmission queue.

For the resolution to succeed:

- The channel must be inactive
- The channel must be in doubt
- The channel type must be sender or server
- The channel definition, local, must exist
- The queue manager must be running, local

18=Display authority

Use the Display authority option to display what actions a user is authorized to perform on a specific IBM MQ object.

For a chosen object, and user, the DSPMQAUT command shows the authorizations the user has to perform actions on an IBM MQ object. If the user is a member of multiple groups, then the command shows the combined authorization of all the groups to the object.

19=Grant authority

Use the Grant authority option to grant the authority to perform actions on IBM MQ objects to another user or group of users.

The GRTMQMAUT command is only available to users in the QMQMADM group. A user in QMQMADM grants authority to other users to perform actions on the IBM MQ objects named in the command either by identifying the users by name, or by granting authority to all users in *PUBLIC.

20=Revoke authority

Use Revoke authority to remove authorization to perform actions on objects from users.

The RVKMQMAUT command is only available to users in the QMQMADM group. A user in the QMQMADM group removes the authority from other users to perform actions on the IBM MQ objects named in the command either by identifying the users by name, or by revoking authority from all users in *PUBLIC.

21=Recover object

Use Recover object to restore damaged objects from information stored in IBM MQ journals.

Recover object uses the Re-create MQ Object command (RCRMQMOBJ) to recover all objects that are damaged named in the command. If an object is not damaged, then no action is performed on that object.

22=Record image

Use Record image to reduce the number of journal receivers required for the recovery of a set of objects, and to minimize recovery time.

The RCDMQMIMG command takes a checkpoint for all the objects that are selected in the command. It synchronizes the current values of the objects in the integrated file system (IFS) with later information about the objects, such as MQPUTs and MQGETs recorded in journal receivers.

When the command completes the objects in the IFS are up to date, and those journal receivers are no longer required to be present to recover the objects. Any disconnected journal receivers can be detached (as long as they are not required to be present to recover other objects).

Setting up communication for IBM MQ for IBM i

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For it to succeed, it is necessary for the connection to be defined and available.

DQM is a remote queuing facility for IBM MQ for IBM i. It provides channel control programs for the IBM MQ for IBM i queue manager which form the interface to communication links, controllable by the system operator.

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For it to succeed, it is necessary for the connection to be defined and available. This section explains how to ensure that the connection is defined and available.

Before a channel can be started, the transmission queue must be defined as described in this section, and must be included in the message channel definition.

You can choose between the following two forms of communication between IBM MQ for IBM i systems:

- "Defining a TCP connection on IBM i" on page 209

  For TCP, a host address can be used, and these connections are set up as described in the *IBM i Communication Configuration Reference*.

  In the TCP environment, each distributed service is allocated a unique TCP address which can be used by remote machines to access the service. The TCP address consists of a host name/number and a port number. All queue managers use such a number to communicate with each other by way of TCP.

- "Receiving on TCP" on page 209

  This form of communication requires the definition of an IBM i SNA logical unit type 6.2 (LU 6.2) that provides the physical link between the IBM i system serving the local queue manager and the system serving the remote queue manager. Refer to the *IBM i Communication Configuration Reference* for details on configuring communications in IBM i.

In addition, where needed, the triggering arrangement must be prepared with the definition of the necessary processes and queues.

Related concepts

"Monitoring and controlling channels on IBM i" on page 195
Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers. Each queue manager has a DQM program for controlling interconnections to compatible remote queue managers.

Related reference

Example configuration - IBM MQ for IBM i

Message channel planning example for IBM MQ for IBM i

Intercommunication jobs on IBM i

Channel states on IBM i

Defining a TCP connection on IBM i

You can define a TCP connection within the channel definition using the Connection Name field.

The channel definition contains a field, CONNECTION NAME, that contains either the TCP network address of the target or the host name (for example ABCHOST). The TCP network address can be in IPv4 dotted decimal form (for example 127.0.0.1) or IPv6 hexadecimal form (for example 2001:DB8:0:0:0:0:0:0). If the CONNECTION NAME is a host name or a name server, the IBM i host table is used to convert the host name into a TCP host address.

A port number is required for a complete TCP address; if this number is not supplied, the default port number 1414 is used. On the initiating end of a connection (sender, requester, and server channel types) it is possible to provide an optional port number for the connection, for example:

```
Connection name 127.0.0.1 (1555)
```

In this case the initiating end attempts to connect to a receiving program at port 1555.

Using the TCP listener backlog option

In TCP, connections are treated incomplete unless three-way handshake takes place between the server and the client. These connections are called outstanding connection requests. A maximum value is set for these outstanding connection requests and can be considered a backlog of requests waiting on the TCP port for the listener to accept the request.

See "Using the TCP listener backlog option" on page 192 for more information, and the specific value for IBM i.

Related concepts

"Receiving on TCP" on page 209

Receiving channel programs are started in response to a startup request from the sending channel. To respond to the startup request, a listener program has to be started to detect incoming network requests and start the associated channel. You start this listener program with the STRMQMLSR command.

Receiving on TCP

Receiving channel programs are started in response to a startup request from the sending channel. To respond to the startup request, a listener program has to be started to detect incoming network requests and start the associated channel. You start this listener program with the STRMQMLSR command.

You can start more than one listener for each queue manager. By default, the STRMQMLSR command uses port 1414 but you can override this value. To override the default setting, add the following statements to the qm.ini file of the selected queue manager. In this example, the listener is required to use port 2500:

```
TCP:
Port=2500
```

The qm.ini file is located in this IFS directory: /QIBM/UserData/mqm/qmgrs/ *queue manager name*.

This new value is read only when the TCP listener is started. If you have a listener already running, this change is not be seen by that program. To use the new value, stop the listener and issue the STRMQMLSR command again. Now, whenever you use the STRMQMLSR command, the listener defaults to the new port.

Alternatively, you can specify a different port number on the STRMQMLSR command. For example:

```
STRMQMLSR MQMNAME( queue manager name ) PORT(2500)
```

This change makes the listener default to the new port for the duration of the listener job.

Using the TCP SO_KEEPALIVE option

If you want to use the SO_KEEPALIVE option (for more information, see "Checking that the other end of the channel is still available" on page 159) you must add the following entry to your queue manager configuration file (qm.ini in the IFS directory, /QIBM/UserData/mqm/qmgrs/ *queue manager name*):

```
TCP:
KeepAlive=yes
```

You must then issue the following command:

```
CFGTCP
```

Select option 3 (Change TCP Attributes). You can now specify a time interval in minutes. You can specify a value in the range 1 through 40320 minutes; the default is 120.

Using the TCP listener backlog option

When receiving on TCP, a maximum number of outstanding connection requests is set. This number can be considered a *backlog* of requests waiting on the TCP port for the listener to accept the request.

The default listener backlog value on IBM i is 255. If the backlog reaches this value, the TCP connection is rejected and the channel is not able to start.

For MCA channels, this results in the channel going into a RETRY state and retrying the connection at a later time.

For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and can retry the connection at a later time.

However, to avoid this error, you can add an entry in the qm.ini file:

```
ListenerBacklog = n
```

This overrides the default maximum number of outstanding requests (255) for the TCP listener.

Note: Some operating systems support a larger value than the default. If necessary, this value can be used to avoid reaching the connection limit.

Defining an LU 6.2 connection on IBM i

Define the LU 6.2 communications details by using a mode name, TP name, and connection name of a fully qualified LU 6.2 connection.

The initiated end of the link must have a routing entry definition to complement this CSI object. Further information about managing work requests from remote LU 6.2 systems is available in the *IBM i Programming: Work Management Guide*.

See the *Multiplatform APPC Configuration Guide* and the following table for information.

Table 24. Settings on the local IBM i system for a remote queue manager platform

Remote platform	**TPNAME**
z/OS or MVS	The same as in the corresponding side information about the remote queue manager.
IBM i	The same as the compare value in the routing entry on the IBM i system.
HP Integrity NonStop Server	The same as the TPNAME specified in the receiver-channel definition.

Table 24. Settings on the local IBM i system for a remote queue manager platform (continued)	
Remote platform	**TPNAME**
UNIX and Linux systems	The invokable Transaction Program defined in the remote LU 6.2 configuration.
Windows	As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.

If you have more than one queue manager on the same computer, ensure that the TPnames in the channel definitions are unique.

Related concepts

"Initiating end (Sender)" on page 211
Use the CRTMQMCHL command to define a channel of transport type *LU62.

"Initiated end (Receiver)" on page 213
Use the CRTMQMCHL command to define the receiving end of the message channel link with transport type *LU62.

Initiating end (Sender)

Use the CRTMQMCHL command to define a channel of transport type *LU62.

Use of the CSI object is optional in IBM MQ for IBM i V5.3 or later.

The initiating end panel is shown in Figure LU 6.2 communication setup panel - initiating end. To obtain the complete panel as shown, press F10 from the first panel.

```
Create Comm Side Information (CRTCSI)

Type choices, press Enter.

Side information . . . . . . . . > WINSDOA1   Name
Library . . . . . . . . . . . >  QSYS    Name, *CURLIB
Remote location . . . . . . . . > WINSDOA1   Name
Transaction program . . . . . . > MQSERIES

Text 'description' . . . . . . .  *BLANK


Additional Parameters

Device . . . . . . . . . . . . .  *LOC     Name, *LOC
Local location . . . . . . . . .  *LOC     Name, *LOC, *NETATR
Mode . . . . . . . . . . . . . .  JSTMOD92   Name, *NETATR
Remote network identifier . . .  *LOC     Name, *LOC, *NETATR, *NONE
Authority . . . . . . . . . . .  *LIBCRTAUT  Name, *LIBCRTAUT, *CHANGE...

Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 34. LU 6.2 communication setup panel - initiating end

Complete the initiating end fields as follows:

Side information
: Give this definition a name that is used to store the side information object to be created, for example, WINSDOA1.

  Note: For LU 6.2, the link between the message channel definition and the communication connection is the **Connection name** field of the message channel definition at the sending end. This field contains the name of the CSI object.

Library
: The name of the library where this definition is stored.

The CSI object must be available in a library accessible to the program serving the message channel, for example, QSYS, QMQM, and QGPL.

If the name is incorrect, missing, or cannot be found then an error occurs on channel startup.

Remote location
Specifies the remote location name with which your program communicates.

In short, this required parameter contains the logical unit name of the partner at the remote system, as defined in the device description that is used for the communication link between the two systems.

The **Remote location** name can be found by issuing the command DSPNETA on the remote system and seeing the default local location name.

Transaction program
Specifies the name (up to 64 characters) of the transaction program on the remote system to be started. It can be a transaction process name, a program name, the channel name, or a character string that matches the **Compare value** in the routing entry.

This parameter is required.

Note: To specify SNA service transaction program names, enter the hexadecimal representation of the service transaction program name. For example, to specify a service transaction program name with a hexadecimal representation of 21F0F0F1, you would enter X'21F0F0F1'.

More information about SNA service transaction program names is in the *SNA Transaction Programmer's Reference* manual for LU Type 6.2.

If the receiving end is another IBM i system, the **Transaction program** name is used to match the CSI object at the sending end with the routing entry at the receiving end. This name must be unique for each queue manager on the target IBM i system. See the **Program to call** parameter under Initiated end (Receiver). See also the **Comparison data: compare value** parameter in the Add Routing Entry panel.

Text description
A description (up to 50 characters) to remind you of the intended use of this connection.

Device
Specifies the name of the device description used for the remote system. The possible values are:

***LOC**
The device is determined by the system.

Device-name
Specify the name of the device that is associated with the remote location.

Local location
Specifies the local location name. The possible values are:

***LOC**
The local location name is determined by the system.

***NETATR**
The LCLLOCNAME value specified in the system network attributes is used.

Local-location-name
Specify the name of your location. Specify the local location if you want to indicate a specific location name for the remote location. The location name can be found by using the DSPNETA command.

Mode
Specifies the mode used to control the session. This name is the same as the Common Programming Interface (CPI)- Communications Mode_Name. The possible values are:

***NETATR**
The mode in the network attributes is used.

BLANK
Eight blank characters are used.

Mode-name
Specify a mode name for the remote location.

Note: Because the mode determines the transmission priority of the communications session, it might be useful to define different modes depending on the priority of the messages being sent; for example MQMODE_HI, MQMODE_MED, and MQMODE_LOW. (You can have more than one CSI pointing to the same location.)

Remote network identifier
Specifies the remote network identifier used with the remote location. The possible values are:

***LOC**
The remote network ID for the remote location is used.

***NETATR**
The remote network identifier specified in the network attributes is used.

***NONE**
The remote network has no name.

Remote-network-id
Specify a remote network ID. Use the DSPNETA command at the remote location to find the name of this network ID. It is the 'local network ID' at the remote location.

Authority
Specifies the authority you are giving to users who do not have specific authority to the object, who are not on an authorization list, and with a group profile that has no specific authority to the object. The possible values are:

***LIBCRTAUT**
Public authority for the object is taken from the CRTAUT parameter of the specified library. This value is determined at create time. If the CRTAUT value for the library changes after the object is created, the new value does not affect existing objects.

***CHANGE**
Change authority allows the user to perform basic functions on the object, however, the user cannot change the object. Change authority provides object operational authority and all data authority.

***ALL**
The user can perform all operations except those operations limited to the owner or controlled by authorization list management authority. The user can control the existence of the object and specify the security for the object, change the object, and perform basic functions on the object. The user can change ownership of the object.

***USE**
Use authority provides object operational authority and read authority.

***EXCLUDE**
Exclude authority prevents the user from accessing the object.

Authorization-list
Specify the name of the authorization list with authority that is used for the side information.

Initiated end (Receiver)
Use the CRTMQMCHL command to define the receiving end of the message channel link with transport type *LU62.

Leave the CONNECTION NAME field blank and ensure that the corresponding details match the sending end of the channel. For details, see Creating a channel.

To enable the initiating end to start the receiving channel, add a routing entry to a subsystem at the initiated end. The subsystem must be the one that allocates the APPC device used in the LU 6.2 sessions. Therefore, it must have a valid communications entry for that device. The routing entry calls the program that starts the receiving end of the message channel.

Use the IBM i commands (for example, ADDRTGE) to define the end of the link that is initiated by a communication session.

The initiated end panel is shown in LU 6.2 communication setup panel - add routing entry.

```
Add Routing Entry (ADDRTGE)

Type choices, press Enter.

Subsystem description . . . . .  QCMN     Name
Library . . . . . . . . . . .   *LIBL    Name, *LIBL, *CURLIB
Routing entry sequence number .  1       1-9999
Comparison data:
Compare value . . . . . . . .  MQSERIES

Starting position . . . . . .  37      1-80
Program to call . . . . . . . .  AMQCRC6B   Name, *RTGDTA
Library . . . . . . . . . . .  QMAS400    Name, *LIBL, *CURLIB
Class . . . . . . . . . . . . .  *SBSD     Name, *SBSD
Library . . . . . . . . . . .  *LIBL     Name, *LIBL, *CURLIB
Maximum active routing steps . .  *NOMAX    0-1000, *NOMAX
Storage pool identifier . . . .  1       1-10



Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

Figure 35. LU 6.2 communication setup panel - initiated end

Subsystem description
The name of your subsystem where this definition resides. Use the IBM i WRKSBSD command to view and update the appropriate subsystem description for the routing entry.

Routing entry sequence number
A unique number in your subsystem to identify this communication definition. You can use values in the range 1 - 9999.

Comparison data: Compare value
A text string to compare with the string received when the session is started by a **Transaction program** parameter, as shown in Figure 1. The character string is derived from the Transaction program field of the sender CSI.

Comparison data: Starting position
The character position in the string where the comparison is to start.

Note: The starting position field is the character position in the string for comparison, and this position is always 37.

Program to call
The name of the program that runs the inbound message program to be called to start the session.

The program, AMQCRC6A, is called for the default queue manager. This program is supplied with IBM MQ for IBM i and sets up the environment and then calls AMQCRS6A.

For additional queue managers:

- Each queue manager has a specific LU 6.2 invokable program located in its library. This program is called AMQCRC6B and is automatically generated when the queue manager is created.
- Each queue manager requires a specific routing entry with unique routing data to be added. This routing data must match the **Transaction program** name supplied by the requesting system (see Initiating end (Sender)).

An example is shown in LU 6.2 communication setup panel - display routing entries:

```
Display Routing Entries
System:  MY400
Subsystem description:  QCMN      Status:  ACTIVE

Type options, press Enter.
5=Display details

Start
Opt  Seq Nbr  Program    Library    Compare Value   Pos
10   *RTGDTA           'QZSCSRVR'     37
20   *RTGDTA           'QZRCSRVR'     37
30   *RTGDTA           'QZHQTRG'     37
50   *RTGDTA           'QVPPRINT'     37
60   *RTGDTA           'QNPSERVR'     37
70   *RTGDTA           'QNMAPINGD'    37
80   QNMAREXECD  QSYS      'AREXECD'     37
90   AMQCRC6A   QMQMBW    'MQSERIES'     37
100   *RTGDTA           'QTFDWNLD'     37
150   *RTGDTA           'QMFRCVR'     37



F3=Exit  F9=Display all detailed descriptions  F12=Cancel
```

Figure 36. LU 6.2 communication setup panel - initiated end

In LU 6.2 communication setup panel - display routing entries, sequence number 90 represents the default queue manager and provides compatibility with configurations from previous releases (that is, V3R2, V3R6, V3R7, and V4R2) of IBM MQ for IBM i. These releases allow one queue manager only. Sequence numbers 92 and 94 represent two additional queue managers called ALPHA and BETA that are created with libraries QMALPHA and QMBETA.

Note: You can have more than one routing entry for each queue manager by using different routing data. These entries give the option of different job priorities depending on the classes used.

Class

The name and library of the class used for the steps started through this routing entry. The class defines the attributes of the routing step's running environment and specifies the job priority. An appropriate class entry must be specified. Use, for example, the WRKCLS command to display existing classes or to create a class. Further information about managing work requests from remote LU 6.2 systems is available in the *IBM i Programming: Work Management Guide*.

Note on Work Management

The AMQCRS6A job is not able to take advantage of the normal IBM i work management features that are documented in Work management because it is not started in the same way as other IBM MQ jobs. To change the runtime properties of the LU62 receiver jobs, you can make one of the following changes:

- Alter the class description that is specified on the routing entry for the AMQCRS6A job
- Change the job description on the communications entry

See the *IBM i Programming: Work Management Guide* for more information about configuring Communication Jobs.

Configuring a queue manager cluster

Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. You can define cluster components, and create and manage clusters.

Before you begin

For an introduction to clustering concepts, see Clusters.

When you are designing your queue manager cluster you have to make some decisions. See Example clusters and Designing clusters.

Related tasks

"Moving a cluster topic definition to a different queue manager" on page 338
For either topic host routed or direct routed clusters, you might need to move a cluster topic definition when decommissioning a queue manager, or because a cluster queue manager has failed or is unavailable for a significant period of time.

Related reference

DELETE TOPIC

Defining components of a cluster

Clusters are composed of queue managers, cluster channels, and cluster queues. You can define cluster queues, and modify some aspects of default cluster objects. You can get configuration and status information about auto-defined channels, and about the relationship between individual cluster-sender channels and transmission queues.

See the following subtopics for information about defining each of the cluster components:

Related concepts

Components of a cluster

Cluster channels

Related tasks

Defining cluster topics

"Setting up a new cluster" on page 227
Follow these instructions to set up the example cluster. Separate instructions describe setting up the cluster on TCP/IP, LU 6.2, and with a single transmission queue or multiple transmission queues. Test the cluster works by sending a message from one queue manager to the other.

"Adding a queue manager to a cluster" on page 237
Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using the single cluster transmission queue `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

Defining cluster queues

A cluster queue is a queue that is hosted by a cluster queue manager and made available to other queue managers in the cluster.Define a cluster queue as a local queue on the cluster queue manager where the queue is hosted. Specify the name of the cluster the queue belongs to.

The following example shows a **runmqsc** command to define a cluster queue with the `CLUSTER` option:

```
DEFINE QLOCAL(Q1) CLUSTER(SALES)
```

A cluster queue definition is advertised to other queue managers in the cluster. The other queue managers in the cluster can put messages to a cluster queue without needing a corresponding remote-queue definition. A cluster queue can be advertised in more than one cluster by using a cluster namelist.

When a queue is advertised, any queue manager in the cluster can put messages to it. To put a message, the queue manager must find out, from the full repositories, where the queue is hosted. Then it adds some routing information to the message and puts the message on a cluster transmission queue.

z/OS A cluster queue can be a queue that is shared by members of a queue-sharing group in IBM MQ for z/OS.

Binding

You can create a cluster in which more than one queue manager hosts an instance of the same cluster queue. Make sure that all the messages in a sequence are sent to the same instance of the queue. You can bind a series of messages to a particular queue by using the MQOO_BIND_ON_OPEN option on the MQOPEN call.

Cluster transmission queues

A queue manager can store messages for other queue managers in a cluster on multiple transmission queues. You can configure a queue manager to store messages on multiple cluster transmission queues in two different ways. If you set the queue manager attribute **DEFCLXQ** to CHANNEL, a different cluster transmission queue is created automatically from SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE for each cluster-sender channel. If you set the CLCHNAME transmission queue option to match one or more cluster-senders channel, the queue manager can store messages for the matching channels on that transmission queue.

Attention: If you are using dedicated SYSTEM.CLUSTER.TRANSMIT.QUEUES with a queue manager that was upgraded from a version of the product earlier than IBM WebSphere MQ 7.5, ensure that the SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE has the SHARE/NOSHARE option set to **SHARE**.

A message for a cluster queue on a different queue manager is placed upon a cluster transmission queue before being sent. A cluster-sender channel transfers the messages from a cluster transmission queue to cluster-receiver channels on other queue managers. By default, one system defined cluster transmission queue holds all the messages that are to be transferred to other cluster queue managers. The queue is called SYSTEM.CLUSTER.TRANSMIT.QUEUE. A queue manager that is part of a cluster can send messages on this cluster transmission queue to any other queue manager in the same cluster.

A definition for the single SYSTEM.CLUSTER.TRANSMIT.QUEUE queue is created by default on every queue manager except on z/OS. On z/OS, the definition can be defined with the supplied sample **CSQ4INSX**.

You can configure a queue manager to transfer messages to other clustered queue managers using multiple transmission queues. You can define additional cluster transmission queues manually, or have the queue manager create the queues automatically.

To have the queues created automatically by the queue manager, change the queue manager attribute DEFCLXQ from SCTQ to CHANNEL. The result is the queue manager creates an individual cluster transmission queue for each cluster-sender channel that is created. The transmission queues are created as permanent dynamic queues from the model queue, SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. The name of each permanent dynamic queue is SYSTEM.CLUSTER.TRANSMIT. *ChannelName*. The name of the cluster-sender channel that each permanent dynamic cluster transmit queue is associated with is set in the local transmission queue attribute CLCHNAME. Messages for remote clustered queue managers are placed on the permanent dynamic cluster transmission queue for the associated cluster-sender channel, rather than on SYSTEM.CLUSTER.TRANSMIT.QUEUE.

To create the cluster transmission queues manually, create a local queue with the USAGE attribute set to XMITQ, and the CLCHNAME attribute set to a generic channel name that resolves to one or more cluster-sender channels; see ClusterChannelName. If you create cluster transmission queues manually, you have the choice of associating the transmission queue with a single cluster-sender channel, or with multiple cluster-sender channels. The CLCHNAME attribute is a generic-name, which means you can place multiple wildcard characters, "*", in the name.

Except for the initial cluster-sender channels that you create manually to connect a queue manager to a full repository, cluster-sender channels are created automatically. They are created automatically when there is a message to transfer to a cluster queue manager. They are created with the same name as the name of the cluster-receiver channel that receives cluster messages for that particular cluster on the destination queue manager.

If you follow a naming convention for cluster-receiver channels, it is possible to define a generic value for CLCHNAME that filters different kinds of cluster messages to different transmission queues. For example, if you follow the naming convention for cluster-receiver channels of *ClusterName*. *QmgrName*, then the generic name *ClusterName*.* filters messages for different clusters onto different transmission queues. You must define the transmission queues manually, and set CLCHNAME in each transmission queue to *ClusterName*.*.

Changes to the association of cluster transmission queues to cluster-sender channels do not take immediate effect. The currently associated transmission queue that a cluster-sender channel is servicing might contain messages that are in the process of being transferred by the cluster-sender channel. Only when no messages on the currently associated transmission queue are being processed by a cluster-sender channel, can the queue manager change the association of the cluster-sender channel a different transmission queue. This can occur either when no messages remain on the transmission queue to be processed by the cluster-sender channel, or when processing of messages is suspended and the cluster-sender channel has no "in-flight" messages. When this happens, any unprocessed messages for the cluster-sender channel are transferred to the newly associated transmission queue, and the association of the cluster-sender channel changes.

You can create a remote queue definition that resolves to a cluster transmission queue. In the definition, queue manager QMX is in the same cluster as the local queue manager, and there is no transmission queue, QMX.

```
DEFINE QREMOTE(A) RNAME(B) RQMNAME(QMX)
```

During queue name resolution, the cluster transmission queue takes precedence over the default transmission queue. A message put to A is stored on the cluster transmission queue and then sent to the remote queue B on QMX.

Queue managers can also communicate with other queue managers that are not part of a cluster. You must define channels and a transmission queue to the other queue manager, in the same way as in a distributed-queuing environment.

Note: Applications must write to queues that resolve to the cluster transmission queue, and must not write directly to the cluster transmission queue.

Auto definition of remote queues

A queue manager in a cluster does not need a remote-queue definition for remote queues in the cluster. The cluster queue manager finds the location of a remote queue from the full repository. It adds routing information to the message and puts it on the cluster transmission queue. IBM MQ automatically creates a definition equivalent to a remote-queue definition so that the message can be sent.

You cannot alter or delete an automatically created remote-queue definition. However, by using the DISPLAY QUEUE **runmqsc** command with the CLUSINFO attribute, you can view all of the local queues on a queue manager as well as all of the cluster queues, including cluster queues on remote queue managers. For example:

```
DISPLAY QUEUE(*) CLUSINFO
```

Related concepts

Cluster queues

Related reference

ClusterChannelName (MQCHAR20)

Working with auto-defined cluster-sender channels

After you introduce a queue manager to a cluster by making its initial CLUSSDR and CLUSRCVR definitions, IBM MQ automatically makes other cluster-sender channel definitions when required to move messages to another queue manager in the cluster. You can view information about auto-defined cluster-sender channels, but you cannot modify them. To modify their behavior, you can use a channel auto-definition exit.

Before you begin

For an introduction to auto-defined channels, see Auto-defined cluster-sender channels.

About this task

Auto-defined cluster-sender channels are created by the cluster as and when needed, and they remain active until they are shut down using the normal disconnect-interval rules.

Cluster sender channels (CLUSSDRs) can be auto-defined both to move application messages and internal cluster administration messages. For example, in a Publish/subscribe cluster (one in which a clustered topic has been defined), channels can be defined between partial repositories to permit exchange of 'proxy subscription' state. When not required (inactive) for an extended period of time auto-defined CLUSSDRs are removed from a partial repository's cache of cluster information and are no longer visible on that queue manager.

On platforms other than z/OS, the OAM (object authority manager) is not aware of the existence of auto-defined cluster-sender channels. If you issue **start**, **stop**, **ping**, **reset**, or **resolve** commands on an auto-defined cluster-sender channel, the OAM checks to see whether you are authorized to perform the same action on the matching cluster-receiver channel.

z/OS On z/OS, you can secure an auto-defined cluster-sender channel in the same way as any other channel.

Procedure

- Display information about the auto-defined channels for a given cluster queue manager.

  You cannot see automatically defined channels using the DISPLAY CHANNEL **runmqsc** command. To see the auto-defined channels use the following command:

  ```
  DISPLAY CLUSQMGR(qMgrName)
  ```

- Display the status of the auto-defined channel for a given CLUSRCVR.

  To display the status of the auto-defined CLUSSDR channel corresponding to a CLUSRCVR channel definition you created, use the following command:

  ```
  DISPLAY CHSTATUS(channelname)
  ```

- Use a channel auto-definition exit to modify the behavior of an auto-defined channel.

  You can use the IBM MQ channel auto-definition exit if you want to write a user exit program to customize a cluster-sender channel or cluster-receiver channel. For example, you can use the channel auto-definition exit in a cluster environment to make any of the following modifications:

  - Tailor communications definitions, that is, SNA LU6.2 names.
  - Add or remove other exits, for example, security exits.
  - Change the names of channel exits.

    The name of the CLUSSDR channel exit is auto-generated from the CLUSRCVR channel definition, and therefore might not be appropriate for your needs - especially if the two ends of the channel are on different platforms.

    The format of exit names is different on different platforms. For example:

    - z/OS On the z/OS platform, the format of the SCYEXIT (*security exit name*) parameter is SCYEXIT('SECEXIT')
    - Windows On Windows platforms, the format of the SCYEXIT (*security exit name*) parameter is SCYEXIT(' *drive*:*path**library* (secexit)')

    Note: z/OS If there is no channel auto-definition exit, the z/OS queue manager derives the CLUSSDR channel exit name from the CLUSRCVR channel definition on the other end of the channel. To derive the z/OS exit name from a non-z/OS name, the following algorithm is used:

    - Exit names on platforms other than z/OS are of the general form *path*/*library* (*function*).
    - If *function* is present, up to eight chars of that are used.
    - Otherwise, up to eight chars of *library* are used.

    For example:

    - `/var/mqm/exits/myExit.so(MsgExit)` converts to `MSGEXIT`
    - `/var/mqm/exits/myExit` converts to `MYEXIT`
    - `/var/mqm/exits/myExit.so(ExitLongName)` converts to `EXITLONG`
  - For queue managers earlier than IBM MQ Version 7, set the **PROPCTL** attribute to a value of `NONE`.

    Each auto-defined cluster-sender channel is based on the corresponding cluster-receiver channel. Before IBM MQ Version 7, the cluster-receiver channel does not have a **PROPCTL** attribute, so this attribute is therefore set to `COMPAT` in the auto-defined cluster-sender channel.

    If the cluster needs to use **PROPCTL** to remove application headers such as RFH2 from messages going from an IBM MQ Version 7 or later queue manager to a queue manager on an earlier version of IBM MQ, you must write a channel auto-definition exit that sets **PROPCTL** to a value of `NONE`.
- Use the channel attribute `LOCLADDR`. to control aspects of addressing.
  - To enable an outbound (TCP) channel to use a particular IP address, port or port range, use the channel attribute `LOCLADDR`. This is useful if you have more than one network card and you want a channel to use a specific one for outbound communications.
  - To specify a virtual IP address on `CLUSSDR` channels, use the IP address from the `LOCLADDR` on a manually defined `CLUSSDR`. To specify the port range, use the port range from the `CLUSRCVR`.
  - If a cluster needs to use `LOCLADDR` to get the outbound communication channels to bind to a specific IP address, you can write a channel auto-definition exit to force the `LOCLADDR` value into any of their automatically defined `CLUSSDR` channels. You must also specify it in the manually defined `CLUSSDR` channel.
  - Put a port number or port range in the `LOCLADDR` of a `CLUSRCVR` channel, if you want all the queue managers in a cluster to use a specific port or range of ports, for all their outbound communications.

  Note: Do not put an IP address in the `LOCLADDR` field of a `CLUSRCVR` channel, unless all queue managers are on the same server. The `LOCLADDR` IP address is propagated to the auto-defined `CLUSSDR` channels of all queue managers that connect using the `CLUSRCVR` channel.

  distributed On distributed platforms, it is possible to set a default local address value that will be used for all sender channels that do not have a local address defined. The default value is defined by setting the MQ_LCLADDR environment variable prior to starting the queue manager. The format of the value matches that of MQSC attribute LOCLADDR.

Related reference

Local Address (LOCLADDR)

Working with default cluster objects

You can alter the default channel definitions in the same way as any other channel definition, by running MQSC or PCF commands. Do not alter the default queue definitions, except for `SYSTEM.CLUSTER.HISTORY.QUEUE`.

For a full list of these objects, see Default cluster objects. The following list only includes those objects that you can change.

SYSTEM.CLUSTER.HISTORY.QUEUE

Each queue manager in a cluster has a local queue called `SYSTEM.CLUSTER.HISTORY.QUEUE`. The `SYSTEM.CLUSTER.HISTORY.QUEUE` is used to store the history of cluster state information for service purposes.

In the default object settings, SYSTEM.CLUSTER.HISTORY.QUEUE is set to PUT (ENABLED). To suppress history collection change the setting to PUT (DISABLED).

SYSTEM.CLUSTER.TRANSMIT.QUEUE

Each queue manager has a definition for a local queue called SYSTEM.CLUSTER.TRANSMIT.QUEUE. SYSTEM.CLUSTER.TRANSMIT.QUEUE is the default transmission queue for all messages to all queues and queue managers that are within clusters. You can change the default transmission queue for each cluster-sender channel to SYSTEM.CLUSTER.TRANSMIT.*ChannelName*, by changing the queue manager attribute DEFXMITQ z/OS, except on z/OS . You cannot delete SYSTEM.CLUSTER.TRANSMIT.QUEUE. It is also used to define authorization checks whether the default transmission queue that is used is SYSTEM.CLUSTER.TRANSMIT.QUEUE or SYSTEM.CLUSTER.TRANSMIT.*ChannelName*.

Related concepts

Default cluster objects

Working with cluster transmission queues and cluster-sender channels

Messages between clustered queue managers are stored on cluster transmission queues and forwarded by cluster-sender channels. At any point in time, a cluster-sender channel is associated with one transmission queue. If you change the configuration of the channel, it might switch to a different transmission queue next time it starts. The processing of this switch is automated, and transactional.

Run the following MQSC command to display the transmission queues that cluster-sender channels are associated with:

```
DISPLAY CHSTATUS(*) WHERE(CHLTYPE EQ CLUSSDR)
```

```
AMQ8417: Display Channel Status details.
CHANNEL(TO.QM2)           CHLTYPE(CLUSSDR)
CONNAME(9.146.163.190(1416))    CURRENT
RQMNAME(QM2)             STATUS(STOPPED)
SUBSTATE( )             XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
```

The transmission queue shown in the saved channel status of a stopped cluster-sender channel might change when the channel starts again. "Selection of default transmission queues by cluster-sender channels" on page 222 describes the process of selecting a default transmission queue; "Selection of manually defined transmission queues by cluster-sender channels" on page 222 describes the process of selecting a manually defined transmission queue.

When any cluster-sender channel starts it rechecks its association with transmission queues. If the configuration of transmission queues, or the queue manager defaults, changes, it might reassociate the channel with a different transmission queue. If the channel restarts with a different transmission queue as a result of a configuration change, a process of transferring messages to the newly associated transmission queue takes place. "How the process to switch cluster-sender channel to a different transmission queue works" on page 223 describes the process of transferring a cluster-sender channel from one transmission queue to another.

The behavior of cluster-sender channels is different to sender and server channels. They remain associated with the same transmission queue until the channel attribute **XMITQ** is altered. If you alter the transmission queue attribute on a sender or server channel, and restart it, messages are not transferred from the old transmission queue to the new one.

Another difference between cluster-sender channels, and sender or server channels, is that multiple cluster-sender channels can open a cluster transmission queue, but only one sender or server channel can open a normal transmission queue. Until Version 7.5, cluster connections shared the single cluster transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE. From Version 7.5 onwards, you have the option of cluster-sender channels not sharing transmission queues. Exclusivity is not enforced; it is an outcome of the configuration. You can configure the path a message takes in a cluster so that it does not share any transmission queues or channels with messages that flow between other applications. See

Clustering: Planning how to configure cluster transmission queues and “Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager” on page 273.

To configure a cluster-sender channel to use a transmission queue other than SYSTEM.CLUSTER.TRANSMIT.QUEUE on z/OS, you need to enable version 8 new function, using the mode of operation (OPMODE) system parameter in the CSQ6SYSP macro.

Selection of default transmission queues by cluster-sender channels

A cluster transmission queue is either a system default queue, with a name that starts with `SYSTEM.CLUSTER.TRANSMIT`, or a manually defined queue. A cluster-sender channel is associated with a cluster transmission queue in one of two ways: by the default cluster transmission queue mechanism, or by manual configuration.

The default cluster transmission queue is set as a queue manager attribute, **DEFCLXQ**. Its value is either `SCTQ` or `CHANNEL`. New and migrated queue managers are set to `SCTQ`. You can alter the value to `CHANNEL`.

If `SCTQ` is set, the default cluster transmission queue is `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. Every cluster-sender channel can open this queue. The cluster-sender channels that do open the queue are the ones that are not associated with manually defined cluster transmission queues.

If `CHANNEL` is set, then the queue manager can create a separate permanent dynamic transmission queue for every cluster-sender channel. Each queue is named `SYSTEM.CLUSTER.TRANSMIT.` *`ChannelName`* and is created from the model queue, `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`. Each cluster-sender channel that is not associated with a manually defined cluster transmission queue is associated with a permanent-dynamic cluster transmission queue. The queue is created by the queue manager when it requires a separate cluster transmission queue for the cluster destination served by this cluster-sender channel, and no queue exists.

Some cluster destinations can be served by cluster-sender channels associated with manually defined transmission queues, and others by the default queue or queues. In the association of cluster-sender channels with transmission queues, the manually defined transmission queues always take precedence over the default transmission queues.

The precedence of cluster transmission queues is illustrated in Figure 37 on page 222. The only cluster-sender channel not associated with a manually defined cluster transmission queue is `CS.QM1`. It is not associated with a manually defined transmission queue, because none of the channel names in the **CLCHNAME** attribute of the transmission queues match `CS.QM1`.

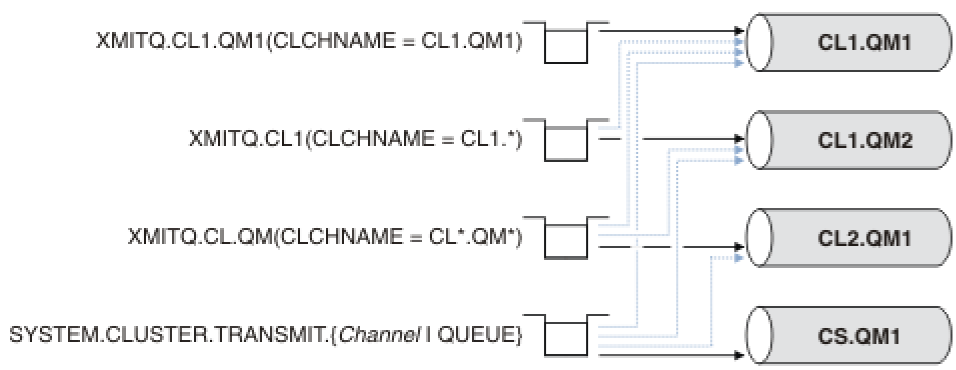

Figure 37. Transmission queue / cluster-sender channel precedence

Selection of manually defined transmission queues by cluster-sender channels

A manually defined queue has the transmission queue attribute **USAGE** attribute set to `XMITQ`, and the cluster channel name attribute **CLCHNAME** set to a specific or generic channel name.

If the name in the **CLCHNAME** queue attribute matches a cluster-sender channel name, the channel is associated with the queue. The name is either an exact match, if the name contains no wildcards, or it the best match, if the name contains wildcards.

If **CLCHNAME** definitions on multiple transmission queues match the same cluster-sender channel, the definitions are said to overlap. To resolve the ambiguity there is an order of precedence between matches. Exact matches always take precedence. Figure 37 on page 222 shows associations between transmission queues and cluster-sender channels. The black arrows show actual associations, and the gray arrows, potential associations. The precedence order of transmission queues in Figure 37 on page 222 is,

XMITQ.CL1.QM1
: The transmission queue XMITQ.CL1.QM1 has its **CLCHNAME** attribute set to CL1.QM1. The definition of the **CLCHNAME** attribute, CL1.QM1, has no wildcards, and takes precedence over any other CLCHNAME attributes, defined on other transmission queues, that match with wildcards. The queue manager stores any cluster message that is to be transferred by the CL1.QM1 cluster-sender channel on the XMITQ.CL1.QM1 transmission queue. The only exception is if multiple transmission queues have their **CLCHNAME** attribute set to CL1.QM1. In that case, the queue manager stores messages for the CL1.QM1 cluster-sender channel on any of those queues. It selects a queue arbitrarily when the channel starts. It might select a different queue when the channel starts again.

XMITQ.CL1
: The transmission queue XMITQ.CL1 has its **CLCHNAME** attribute set to CL1.*. The definition of the **CLCHNAME** attribute, CL1.*, has one trailing wildcard, which matches the name of any cluster-sender channel that starts with CL1.. The queue manager stores any cluster message that is to be transferred by any cluster-sender channel whose name begins with CL1. on the transmission queue XMITQ.CL1, unless there is a transmission queue with a more specific match, such as the queue XMITQ.CL1.QM1. One trailing wildcard makes the definition less specific than a definition with no wildcards, and more specific than a definition with multiple wildcards, or wildcards that are followed by more trailing characters.

XMITQ.CL.QM
: XMITQ.CL.QM is the name of the transmission queue with its **CLCHNAME** attribute set to CL*.QM*. The definition of CL*.QM* has two wildcards, which match the name of any cluster-sender channel that starts with CL., and either includes or ends with QM. The match is less specific than a match with one wildcard.

SYSTEM.CLUSTER.TRANSMIT. *channelName* |QUEUE
: If no transmission queue has a **CLCHNAME** attribute that matches the name of the cluster-sender channel that the queue manager is to use, then the queue manager uses the default cluster transmission queue. The default cluster transmission queue is either the single system cluster transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE, or a system cluster transmission queue that the queue manager created for a specific cluster-sender channel, SYSTEM.CLUSTER.TRANSMIT. *channelName*. Which queue is the default depends on the setting of the queue manager **DEFXMITQ** attribute.

Tip: Unless you have a clear need for overlapping definitions, avoid them as they can lead to complicated configurations that are hard to understand.

How the process to switch cluster-sender channel to a different transmission queue works

To change the association of cluster-sender channels with cluster transmission queues, change the **CLCHNAME** parameter of any transmission queue or the queue manager parameter **DEFCLXQ** at any time. Nothing happens immediately. Changes occur only when a channel starts. When it starts, it checks whether to continue forwarding messages from the same transmission queue. Three kinds of change alter the association of a cluster-sender channel with a transmission queue.

1. Redefining the **CLCHNAME** parameter of the transmission queue the cluster-sender channel is currently associated with to be less specific or blank, or deleting the cluster transmission queue when the channel is stopped.

Some other cluster transmission queue might now be a better match for the channel name. Or, if no other transmission queues match the name of the cluster-sender channel, the association must revert to the default transmission queue.

2. Redefining the **CLCHNAME** parameter of any other cluster transmission queue, or adding a cluster transmission queue.

   The **CLCHNAME** parameter of another transmission queue might now be a better match for the cluster-sender channel than the transmission queue the cluster-sender channel is currently associated with. If the cluster-sender channel is currently associated with a default cluster transmission queue, it might become associated with a manually defined cluster transmission queue.

3. If the cluster-sender channel is currently associated with a default cluster transmission queue, changing the **DEFCLXQ** queue manager parameter.

If the association of a cluster-sender channel changes, when the channel starts it switches its association to the new transmission queue. During the switch, it ensures that no messages are lost. Messages are transferred to the new transmission queue in the order in which the channel would transfer the messages to the remote queue manager.

Remember: In common with any forwarding of messages in a cluster, you must put messages into groups to ensure that messages that must be delivered in order are delivered in order. On rare occasions, messages can get out of order in a cluster.

The switch process goes through the following transactional steps. If the switch process is interrupted, the current transactional step is resumed when the channel restarts again.

Step 1 - Process messages from the original transmission queue

The cluster-sender channel is associated with the new transmission queue, which it might share with other cluster-sender channels. Messages for the cluster-sender channel continue to be placed on the original transmission queue. A transitional switch process transfers messages from the original transmission queue onto the new transmission queue. The cluster-sender channel forwards the messages from the new transmission queue to the cluster-receiver channel. The channel status shows the cluster-sender channel still associated with the old transmission queue.

The switch process continues to transfer newly arrived messages as well. This step continues until the number of remaining messages to be forwarded by the switch process reaches zero. When the number of messages reaches zero, the procedure moves onto step 2.

During step 1, disk activity for the channel increases. Persistent messages are committed off the first transmission queue and onto the second transmission queue. This disk activity is in addition to messages being committed when they are placed on and removed from the transmission queue as part of transferring the messages normally. Ideally, no messages arrive during the switching process, so the transition can take place as quickly as possible. If messages do arrive, they are processed by the switch process.

Step 2 - Process messages from the new transmission queue

As soon as no messages remain on the original transmission queue for the cluster-sender channel, new messages are placed directly on the new transmission queue. The channel status shows the cluster-sender channel is associated with the new transmission queue. The following message is written to the queue manager error log: " AMQ7341 `The transmission queue for channel` *`ChannelName`* `is` *`QueueName`* ."

Multiple cluster transmission queues and cluster transmission queue attributes

You have a choice of forwarding cluster messages to different queue managers storing the messages on a single cluster transmission queue, or multiple queues. With one queue, you have one set of cluster transmission queue attributes to set and query; with multiple queues, you have multiple sets. For some attributes, having multiple sets is an advantage: for example querying queue depth tells you how many messages are waiting to be forwarded by one or a set of channels, rather than by all channels. For other attributes, having multiple sets is a disadvantage: for example, you probably do not want to configure the same access permissions for every cluster transmission queue. For this reason, access permissions are always checked against the profile for SYSTEM.CLUSTER.TRANSMIT.QUEUE, and not against profiles for a particular cluster transmission queue. If you want to apply more granular security checks, see Access control and multiple cluster transmission queues.

Multiple cluster-sender channels and multiple transmission queues

A queue manager stores a message on a cluster transmission queue before forwarding it on a cluster-sender channel. It selects a cluster-sender channel that is connected to the destination for the message. It might have a choice of cluster-sender channels that all connect to the same destination. The destination might be the same physical queue, connected by multiple cluster-sender channels to a single queue manager. The destination might also be many physical queues with the same queue name, hosted on different queue managers in the same cluster. Where there is a choice of cluster-sender channels connected to a destination, the workload balancing algorithm chooses one. The choice depends on a number of factors; see The cluster workload management algorithm.

In Figure 38 on page 226, CL1.QM1, CL1.QM2 and CS.QM1 are all channels that might lead to the same destination. For example, if you define Q1 in CL1 on QM1 and QM2 then CL1.QM1 and CL1.QM2 both provide routes to the same destination, Q1, on two different queue managers. If the channel CS.QM1 is also in CL1, it too is a channel that a message for Q1 can take. The cluster membership of CS.QM1 might be defined by a cluster namelist, which is why the channel name does not include a cluster name in its construction. Depending on the workload balancing parameters, and the sending application, some messages for Q1 might be placed on each of the transmission queues, XMITQ.CL1.QM1, XMITQ.CL1 and SYSTEM.CLUSTER.TRANSMIT.CS.QM1.

If you intend to separate out message traffic, so that messages for the same destination do not share queues or channels with messages for different destinations, you must consider how to divide traffic onto different cluster-sender channels first, and then how to separate messages for a particular channel onto a different transmission queue. Cluster queues on the same cluster, on the same queue manager, normally share the same cluster channels. Defining multiple cluster transmission queues alone is not sufficient to separate cluster message traffic onto different queues. Unless you separate messages for different destination queues onto different channels, the messages share the same cluster transmission queue.

A straightforward way to separate the channels that messages take, is to create multiple clusters. On any queue manager in each cluster, define only one cluster queue. Then, if you define a different cluster-receiver channel for each cluster/queue manager combination, the messages for each cluster queue do not share a cluster channel with messages for other cluster queues. If you define separate transmission queues for the cluster channels, the sending queue manager stores messages for only one cluster queue on each transmission queue. For example, if you want two cluster queues not to share resources, you can either place them in different clusters on the same queue manager, or on different queue managers in the same cluster.

The choice of cluster transmission queue does not affect the workload balancing algorithm. The workload balancing algorithm chooses which cluster-sender channel to forward a message. It places the message on the transmission queue that is serviced by that channel. If the workload balancing algorithm is called on to choose again, for instance if the channel stops, it might be able to select a different channel to forward the message. If it does choose a different channel, and the new channel forwards messages from a different cluster transmission queue, the workload balancing algorithm transfers the message to the other transmission queue.

In Figure 38 on page 226, two cluster-sender channels, CS.QM1 and CS.QM2, are associated with the default system transmission queue. When the workload balancing algorithm stores a message on SYSTEM.CLUSTER.TRANSMIT.QUEUE, or any other cluster transmission queue, the name of the cluster-sender channel that is to forward the message is stored in the correlation ID of the message. Each channel forwards just those messages that match the correlation ID with the channel name.

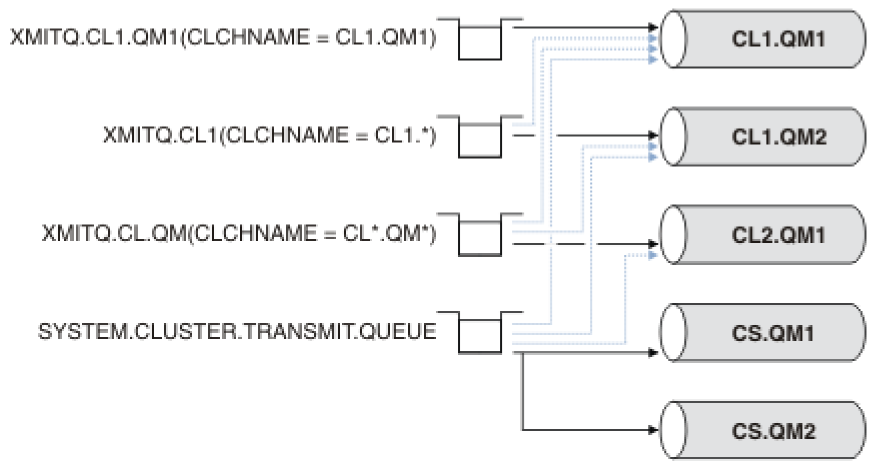

Figure 38. Multiple cluster sender channels

If CS.QM1 stops, the messages on the transmission queue for that cluster-sender channel are examined. Those messages that can be forwarded by another channel are reprocessed by the workload balancing algorithm. Their correlation ID is reset to an alternative cluster-sender channel name. If the alternative cluster-sender channel is CS.QM2, the message remains on SYSTEM.CLUSTER.TRANSMIT.QUEUE. If the alternative channel is CL1.QM1, the workload balancing algorithm transfers the message to XMITQ.CL1.QM1. When the cluster-sender channel restarts, new messages, and messages that were not flagged for a different cluster-sender channel, are transferred by the channel again.

You might change the association between transmission queues and cluster-sender channels on a running system. You might change a CLCHNAME parameter on a transmission queue, or, change the **DEFCLXQ** queue manager parameter. When a channel that is affected by the change restarts, it starts the transmission queue switching process; see "How the process to switch cluster-sender channel to a different transmission queue works" on page 223.

The process to switch the transmission queue starts when the channel is restarted. The workload rebalancing process starts when the channel is stopped. The two process can run in parallel.

The simple case is when stopping a cluster-sender channel does not cause the rebalancing process to change the cluster-sender channel that is to forward any messages on the queue. This case is when no other cluster-sender channel can forward the messages to the correct destination. With no alternative cluster-sender channel to forward the messages to their destination, the messages remain flagged for the same cluster-sender channel after the cluster-sender channel stops. When the channel starts, if a switch is pending, the switching processes moves the messages to a different transmission queue where they are processed by the same cluster-sender channel.

The more complex case is where more than one cluster-sender channel can process some messages to the same destination. You stop and restart the cluster-sender channel to trigger the transmission queue switch. In many cases, by the time you restart the channel, the workload balancing algorithm has already moved messages from the original transmission queue to different transmission queues served by different cluster-sender channels. Only those messages that cannot be forwarded by a different cluster-sender channel remain to be transferred to the new transmission queue. In some cases, if the channel is restarted quickly, some messages that could be transferred by the workload balancing algorithm remain. In which case some remaining messages are switched by the workload balancing process, and some by the process of switching the transmission queue.

Related concepts

Cluster channels

Clustering: Application isolation using multiple cluster transmission queues

"Calculating the size of the log" on page 463

Estimating the size of log a queue manager needs.

Related tasks

Clustering: Planning how to configure cluster transmission queues

"Creating two-overlapping clusters with a gateway queue manager" on page 263
Follow the instructions in the task to construct overlapping clusters with a gateway queue manager. Use the clusters as a starting point for the following examples of isolating messages to one application from messages to other applications in a cluster.

"Adding a queue manager to a cluster: separate transmission queues" on page 240
Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using multiple cluster transmission queues.

"Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager" on page 270
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses an additional cluster transmission queue to separate message traffic to a single queue manager in a cluster.

"Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager" on page 273
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses an additional cluster to isolate the messages to a particular cluster queue.

Setting up a new cluster

Follow these instructions to set up the example cluster. Separate instructions describe setting up the cluster on TCP/IP, LU 6.2, and with a single transmission queue or multiple transmission queues. Test the cluster works by sending a message from one queue manager to the other.

Before you begin

- Instead of following these instructions, you can use one of the wizards supplied with MQ Explorer to create a cluster like the one created by this task. Right-click the Queue Manager Clusters folder, then click **New** > **Queue Manager Cluster**, and follow the instructions given in the wizard.
- For background information to aid your understanding of the steps taken to set up a cluster, see "Defining cluster queues" on page 216, Cluster channels and Listeners.

About this task

You are setting up a new IBM MQ network for a chain store. The store has two branches, one in London and one in New York. The data and applications for each store are hosted by systems running separate queue managers. The two queue managers are called `LONDON` and `NEWYORK`. The inventory application runs on the system in New York, connected to queue manager `NEWYORK`. The application is driven by the arrival of messages on the `INVENTQ` queue, hosted by `NEWYORK`. The two queue managers, `LONDON` and `NEWYORK`, are to be linked in a cluster called `INVENTORY` so that they can both put messages to the `INVENTQ`.

Figure 39 on page 228 shows what this cluster looks like.

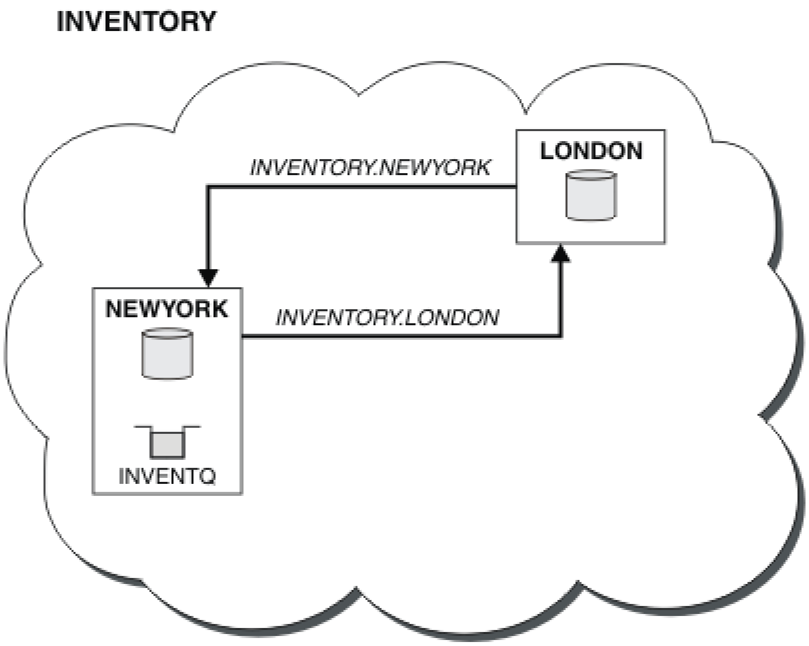

Figure 39. The INVENTORY cluster with two queue managers

You can configure each queue manager in the cluster to send messages to other queue managers in the cluster using different cluster transmission queues.

The instructions to set up the cluster vary a little by transport protocol , number of transmission queues, or platform. You have a choice of three combinations. The verification procedure remains the same for all combinations.

Procedure

- “Setting up a cluster using TCP/IP with a single transmission queue per queue manager” on page 229
- “Setting up a cluster on TCP/IP using multiple transmission queues per queue manager” on page 231
- “Setting up a cluster using LU 6.2 on z/OS” on page 234
- “Verifying the cluster” on page 236

Results

Figure 39 on page 228 shows the `INVENTORY` cluster setup by this task.

Clearly, `INVENTORY` is a small cluster. However, it is useful as a proof of concept. The important thing to understand about this cluster is the scope it offers for future enhancement.

Related concepts
Clusters
Comparison of clustering and distributed queuing
Components of a cluster
Related tasks
“Configuring a queue manager cluster” on page 215

Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. You can define cluster components, and create and manage clusters.

Setting up a cluster using TCP/IP with a single transmission queue per queue manager

Before you begin

- The queue manager attribute, **DEFCLXQ**, must be left as its default value, SCTQ.

About this task

Follow these steps to set up a cluster on AIX, HP-UX, IBM i, Linux, Solaris, and Windows using the transport protocol TCP/IP. z/OS On z/OS, you must follow the instructions in "Defining a TCP connection on z/OS" on page 608 to set up the TCP/IP connection , rather than defining the listeners in step "4" on page 230. Otherwise, the steps are the same for z/OS, but error messages are written to the console, rather than to the queue manager error log.

Procedure

1. Decide on the organization of the cluster and its name.

   You decided to link the two queue managers, LONDON and NEWYORK, into a cluster. A cluster with only two queue managers offers only marginal benefit over a network that is to use distributed queuing. It is a good way to start and it provides scope for future expansion. When you open new branches of your store, you are able to add the new queue managers to the cluster easily. Adding new queue managers does not disrupt the existing network; see "Adding a queue manager to a cluster" on page 237.

   For the time being, the only application you are running is the inventory application. The cluster name is INVENTORY.
2. Decide which queue managers are to hold full repositories.

   In any cluster you must nominate at least one queue manager, or preferably two, to hold full repositories. In this example, there are only two queue managers, LONDON and NEWYORK, both of which hold full repositories.

   a. You can perform the remaining steps in any order.
   b. As you proceed through the steps, warning messages might be written to the queue-manager log. The messages are a result of missing definitions that you have yet to add.

   ```
   Examples of the responses to the commands are shown in a box
   like this after each step in this task.
   These examples show the responses returned by  IBM MQ for AIX.
   The responses vary on other platforms.
   ```

   c. Before proceeding with these steps, make sure that the queue managers are started.
3. Alter the queue-manager definitions to add repository definitions.

   On each queue manager that is to hold a full repository, alter the local queue-manager definition, using the ALTER QMGR command and specifying the REPOS attribute:

   ```
   ALTER QMGR REPOS(INVENTORY)
   ```

   ```
   1 : ALTER QMGR REPOS(INVENTORY)
   AMQ8005: Websphere MQ queue manager changed.
   ```

   For example, if you enter:

a. `runmqsc LONDON`

b. `ALTER QMGR REPOS(INVENTORY)`

LONDON is changed to a full repository.

4. Define the listeners.

   Define a listener that accepts network requests from other queue managers for every queue manager in the cluster. On the LONDON queue managers, issue the following command:

   ```
   DEFINE LISTENER(LONDON_LS) TRPTYPE(TCP) CONTROL(QMGR)
   ```

   The CONTROL attribute ensures that the listener starts and stops when the queue manager does.

   The listener is not started when it is defined, so it must be manually started the first time with the following MQSC command:

   ```
   START LISTENER(LONDON_LS)
   ```

   Issue similar commands for all the other queue managers in the cluster, changing the listener name for each one.

   There are several ways to define these listeners, as shown in Listeners.

5. Define the CLUSRCVR channel for the LONDON queue manager.

   On every queue manager in a cluster, you define a cluster-receiver channel on which the queue manager can receive messages. See Cluster-receiver channel: CLUSRCVR . The CLUSRCVR channel defines the connection name of the queue manager. The connection name is stored in the repositories, where other queue managers can refer to it. The CLUSTER keyword shows the availability of the queue manager to receive messages from other queue managers in the cluster.

   In this example the channel name is `INVENTORY.LONDON`, and the connection name (`CONNAME`) is the network address of the machine the queue manager resides on, which is `LONDON.CHSTORE.COM`. The network address can be entered as an alphanumeric DNS host name, or an IP address in either in IPv4 dotted decimal form. For example, `192.0.2.0`, or IPv6 hexadecimal form; for example `2001:DB8:0204:acff:fe97:2c34:fde0:3485`. The port number is not specified, so the default port (1414) is used.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-receiver channel for queue manager LONDON')
   ```

   ```
   1 : DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-receiver channel for queue manager LONDON')
   AMQ8014: Websphere MQ channel created.
   07/09/98 12:56:35 No repositories for cluster 'INVENTORY'
   ```

6. Define the CLUSRCVR channel for the NEWYORK queue manager.

   If the channel listener is using the default port, typically 1414, and the cluster does not include a queue manager on z/OS, you can omit the CONNAME

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-receiver channel for queue manager NEWYORK')
   ```

7. Define the CLUSSDR channel on the LONDON queue manager.

   You manually define a CLUSSDR channel from every full repository queue manager to every other full repository queue manager in the cluster. See Cluster-sender channel: CLUSSDR . In this case, there are only two queue managers, both of which hold full repositories. They each need a manually-defined CLUSSDR channel that points to the CLUSRCVR channel defined at the other queue manager. The channel names given on the CLUSSDR definitions must match the channel names on the corresponding CLUSRCVR definitions. When a queue manager has definitions for both a cluster-receiver channel and a cluster-sender channel in the same cluster, the cluster-sender channel is started.

```
DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('TCP Cluster-sender channel from LONDON to repository at NEWYORK')
```

```
1 : DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('TCP Cluster-sender channel from LONDON to repository at NEWYORK')
AMQ8014: Websphere MQ channel created.
07/09/98 13:00:18  Channel program started.
```

8. Define the CLUSSDR channel on the NEWYORK queue manager.

```
DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('TCP Cluster-sender channel from NEWYORK to repository at LONDON')
```

9. Define the cluster queue INVENTQ

   Define the INVENTQ queue on the NEWYORK queue manager, specifying the CLUSTER keyword.

```
DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
```

```
1 : DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
AMQ8006: Websphere MQ queue created.
```

   The CLUSTER keyword causes the queue to be advertised to the cluster. As soon as the queue is defined it becomes available to the other queue managers in the cluster. They can send messages to it without having to make a remote-queue definition for it.

   All the definitions are complete. On all platforms, start a listener program on each queue manager. The listener program waits for incoming network requests and starts the cluster-receiver channel when it is needed.

Setting up a cluster on TCP/IP using multiple transmission queues per queue manager

About this task

Follow these steps to set up a cluster on AIX, HP-UX, IBM i, Linux, Solaris, and Windows using the transport protocol TCP/IP. The repository queue managers are configured to use a different cluster transmission queue to send messages to each other, and to other queue managers in the cluster. If you add queue managers to the cluster that are also to use different transmission queues, follow the task, "Adding a queue manager to a cluster: separate transmission queues" on page 240.

Procedure

1. Decide on the organization of the cluster and its name.

   You decided to link the two queue managers, LONDON and NEWYORK, into a cluster. A cluster with only two queue managers offers only marginal benefit over a network that is to use distributed queuing. It is a good way to start and it provides scope for future expansion. When you open new branches of your store, you are able to add the new queue managers to the cluster easily. Adding new queue managers does not disrupt the existing network; see "Adding a queue manager to a cluster" on page 237.

   For the time being, the only application you are running is the inventory application. The cluster name is INVENTORY.

2. Decide which queue managers are to hold full repositories.

   In any cluster you must nominate at least one queue manager, or preferably two, to hold full repositories. In this example, there are only two queue managers, LONDON and NEWYORK, both of which hold full repositories.

   a. You can perform the remaining steps in any order.
   b. As you proceed through the steps, warning messages might be written to the queue-manager log. The messages are a result of missing definitions that you have yet to add.

   ```
   Examples of the responses to the commands are shown in a box
   like this after each step in this task.
   These examples show the responses returned by  IBM MQ for AIX.
   The responses vary on other platforms.
   ```

   c. Before proceeding with these steps, make sure that the queue managers are started.
3. Alter the queue-manager definitions to add repository definitions.

   On each queue manager that is to hold a full repository, alter the local queue-manager definition, using the ALTER QMGR command and specifying the REPOS attribute:

   ```
   ALTER QMGR REPOS(INVENTORY)
   ```

   ```
   1 : ALTER QMGR REPOS(INVENTORY)
   AMQ8005: Websphere MQ queue manager changed.
   ```

   For example, if you enter:

   a. runmqsc LONDON
   b. ALTER QMGR REPOS(INVENTORY)

   LONDON is changed to a full repository.
4. Alter the queue-manager definitions to create separate cluster transmission queues for each destination.

   ```
   ALTER QMGR DEFCLXQ(CHANNEL)
   ```

   On each queue manager that you add to the cluster decide whether to use separate transmission queues or not. See the topics, "Adding a queue manager to a cluster" on page 237 and "Adding a queue manager to a cluster: separate transmission queues" on page 240.
5. Define the listeners.

   Define a listener that accepts network requests from other queue managers for every queue manager in the cluster. On the LONDON queue managers, issue the following command:

   ```
   DEFINE LISTENER(LONDON_LS) TRPTYPE(TCP) CONTROL(QMGR)
   ```

   The CONTROL attribute ensures that the listener starts and stops when the queue manager does.

   The listener is not started when it is defined, so it must be manually started the first time with the following MQSC command:

   ```
   START LISTENER(LONDON_LS)
   ```

   Issue similar commands for all the other queue managers in the cluster, changing the listener name for each one.

   There are several ways to define these listeners, as shown in Listeners.

6. Define the CLUSRCVR channel for the LONDON queue manager.

   On every queue manager in a cluster, you define a cluster-receiver channel on which the queue manager can receive messages. See Cluster-receiver channel: CLUSRCVR . The CLUSRCVR channel defines the connection name of the queue manager. The connection name is stored in the repositories, where other queue managers can refer to it. The CLUSTER keyword shows the availability of the queue manager to receive messages from other queue managers in the cluster.

   In this example the channel name is INVENTORY.LONDON, and the connection name (CONNAME) is the network address of the machine the queue manager resides on, which is LONDON.CHSTORE.COM. The network address can be entered as an alphanumeric DNS host name, or an IP address in either in IPv4 dotted decimal form. For example, 192.0.2.0, or IPv6 hexadecimal form; for example 2001:DB8:0204:acff:fe97:2c34:fde0:3485. The port number is not specified, so the default port (1414) is used.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-receiver channel for queue manager LONDON')
   ```

   ```
   1 : DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-receiver channel for queue manager LONDON')
   AMQ8014: Websphere MQ channel created.
   07/09/98 12:56:35 No repositories for cluster 'INVENTORY'
   ```

7. Define the CLUSRCVR channel for the NEWYORK queue manager.

   If the channel listener is using the default port, typically 1414, and the cluster does not include a queue manager on z/OS, you can omit the CONNAME

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-receiver channel for queue manager NEWYORK')
   ```

8. Define the CLUSSDR channel on the LONDON queue manager.

   You manually define a CLUSSDR channel from every full repository queue manager to every other full repository queue manager in the cluster. See Cluster-sender channel: CLUSSDR . In this case, there are only two queue managers, both of which hold full repositories. They each need a manually-defined CLUSSDR channel that points to the CLUSRCVR channel defined at the other queue manager. The channel names given on the CLUSSDR definitions must match the channel names on the corresponding CLUSRCVR definitions. When a queue manager has definitions for both a cluster-receiver channel and a cluster-sender channel in the same cluster, the cluster-sender channel is started.

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-sender channel from LONDON to repository at NEWYORK')
   ```

   ```
   1 : DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-sender channel from LONDON to repository at NEWYORK')
   AMQ8014: Websphere MQ channel created.
   07/09/98 13:00:18  Channel program started.
   ```

9. Define the CLUSSDR channel on the NEWYORK queue manager.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('TCP Cluster-sender channel from NEWYORK to repository at LONDON')
   ```

10. Define the cluster queue INVENTQ

Define the INVENTQ queue on the NEWYORK queue manager, specifying the CLUSTER keyword.

```
DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
```

```
1 : DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
AMQ8006: Websphere MQ queue created.
```

The CLUSTER keyword causes the queue to be advertised to the cluster. As soon as the queue is defined it becomes available to the other queue managers in the cluster. They can send messages to it without having to make a remote-queue definition for it.

All the definitions are complete. On all platforms, start a listener program on each queue manager. The listener program waits for incoming network requests and starts the cluster-receiver channel when it is needed.

Setting up a cluster using LU 6.2 on z/OS

Procedure

1. Decide on the organization of the cluster and its name.

   You decided to link the two queue managers, LONDON and NEWYORK, into a cluster. A cluster with only two queue managers offers only marginal benefit over a network that is to use distributed queuing. It is a good way to start and it provides scope for future expansion. When you open new branches of your store, you are able to add the new queue managers to the cluster easily. Adding new queue managers does not disrupt the existing network; see "Adding a queue manager to a cluster" on page 237.

   For the time being, the only application you are running is the inventory application. The cluster name is INVENTORY.
2. Decide which queue managers are to hold full repositories.

   In any cluster you must nominate at least one queue manager, or preferably two, to hold full repositories. In this example, there are only two queue managers, LONDON and NEWYORK, both of which hold full repositories.

   a. You can perform the remaining steps in any order.
   b. As you proceed through the steps, warning messages might be written the z/OS system console. The messages are a result of missing definitions that you have yet to add.
   c. Before proceeding with these steps, make sure that the queue managers are started.
3. Alter the queue-manager definitions to add repository definitions.

   On each queue manager that is to hold a full repository, alter the local queue-manager definition, using the ALTER QMGR command and specifying the REPOS attribute:

   ```
   ALTER QMGR REPOS(INVENTORY)
   ```

   ```
   1 : ALTER QMGR REPOS(INVENTORY)
   AMQ8005: Websphere MQ queue manager changed.
   ```

   For example, if you enter:

   a. `runmqsc LONDON`
   b. `ALTER QMGR REPOS(INVENTORY)`

   LONDON is changed to a full repository.
4. Define the listeners.

z/OS See The channel initiator on z/OS and “Receiving on LU 6.2” on page 611.

The listener is not started when it is defined, so it must be manually started the first time with the following MQSC command:

```
START LISTENER(LONDON_LS)
```

Issue similar commands for all the other queue managers in the cluster, changing the listener name for each one.

5. Define the CLUSRCVR channel for the LONDON queue manager.

   On every queue manager in a cluster, you define a cluster-receiver channel on which the queue manager can receive messages. See Cluster-receiver channel: CLUSRCVR . The CLUSRCVR channel defines the connection name of the queue manager. The connection name is stored in the repositories, where other queue managers can refer to it. The CLUSTER keyword shows the availability of the queue manager to receive messages from other queue managers in the cluster.

```
DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(LU62)
CONNAME(LONDON.LUNAME) CLUSTER(INVENTORY)
MODENAME('#INTER') TPNAME('MQSERIES')
DESCR('LU62 Cluster-receiver channel for queue manager LONDON')
```

```
1 : DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) TRPTYPE(LU62)
CONNAME(LONDON.LUNAME) CLUSTER(INVENTORY)
MODENAME('#INTER') TPNAME('MQSERIES')
DESCR('LU62 Cluster-receiver channel for queue manager LONDON')
AMQ8014: Websphere MQ channel created.
07/09/98 12:56:35 No repositories for cluster 'INVENTORY'
```

6. Define the CLUSRCVR channel for the NEWYORK queue manager.

```
DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSRCVR) TRPTYPE(LU62)
CONNAME(NEWYORK.LUNAME) CLUSTER(INVENTORY)
MODENAME('#INTER') TPNAME('MQSERIES')
DESCR('LU62 Cluster-receiver channel for queue manager NEWYORK')
```

7. Define the CLUSSDR channel on the LONDON queue manager.

   You manually define a CLUSSDR channel from every full repository queue manager to every other full repository queue manager in the cluster. See Cluster-sender channel: CLUSSDR . In this case, there are only two queue managers, both of which hold full repositories. They each need a manually-defined CLUSSDR channel that points to the CLUSRCVR channel defined at the other queue manager. The channel names given on the CLUSSDR definitions must match the channel names on the corresponding CLUSRCVR definitions. When a queue manager has definitions for both a cluster-receiver channel and a cluster-sender channel in the same cluster, the cluster-sender channel is started.

```
DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(LU62)
CONNAME(CPIC) CLUSTER(INVENTORY)
DESCR('LU62 Cluster-sender channel from LONDON to repository at NEWYORK')
```

```
1 : DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(LU62)
CONNAME(NEWYORK.LUNAME) CLUSTER(INVENTORY)
MODENAME('#INTER') TPNAME('MQSERIES')
DESCR('LU62 Cluster-sender channel from LONDON to repository at NEWYORK')
AMQ8014: Websphere MQ channel created.
07/09/98 13:00:18  Channel program started.
```

8. Define the CLUSSDR channel on the NEWYORK queue manager.

```
DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSSDR) TRPTYPE(LU62)
```

```
CONNAME(LONDON.LUNAME) CLUSTER(INVENTORY)
DESCR('LU62 Cluster-sender channel from NEWYORK to repository at LONDON')
```

9. Define the cluster queue INVENTQ

   Define the INVENTQ queue on the NEWYORK queue manager, specifying the CLUSTER keyword.

   ```
   DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
   ```

   ```
   1 : DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
   AMQ8006: Websphere MQ queue created.
   ```

   The CLUSTER keyword causes the queue to be advertised to the cluster. As soon as the queue is defined it becomes available to the other queue managers in the cluster. They can send messages to it without having to make a remote-queue definition for it.

   All the definitions are complete. On all platforms, start a listener program on each queue manager. The listener program waits for incoming network requests and starts the cluster-receiver channel when it is needed.

Verifying the cluster

About this task

You can verify the cluster in one or more of these ways:

1. Running administrative commands to display cluster and channel attributes.
2. Run the sample programs to send and receive messages on a cluster queue.
3. Write your own programs to send a request message to a cluster queue and reply with a response messages to an non-clustered reply queue.

Procedure

Issue DISPLAY **runmqsc** commands to verify the cluster.

The responses you see ought to be like the responses in the steps that follow.

1. From the NEWYORK queue manager, run the **DISPLAY CLUSQMGR** command:

   ```
   dis clusqmgr(*)
   ```

   ```
   1 : dis clusqmgr(*)
   AMQ8441: Display Cluster Queue Manager details.
   CLUSQMGR(NEWYORK)       CLUSTER(INVENTORY)
   CHANNEL(INVENTORY.NEWYORK)
   AMQ8441: Display Cluster Queue Manager details.
   CLUSQMGR(LONDON)       CLUSTER(INVENTORY)
   CHANNEL(INVENTORY.LONDON)
   ```

2. From the NEWYORK queue manager, run the **DISPLAY CHANNEL STATUS** command:

   ```
   dis chstatus(*)
   ```

```
1 : dis chstatus(*)
AMQ8417: Display Channel Status details.
CHANNEL(INVENTORY.NEWYORK)  XMITQ( )
CONNAME(192.0.2.0)        CURRENT
CHLTYPE(CLUSRCVR)         STATUS(RUNNING)
RQMNAME(LONDON)
AMQ8417: Display Channel Status details.
CHANNEL(INVENTORY.LONDON) XMITQ(SYSTEM.CLUSTER.TRANSMIT.INVENTORY.LONDON)
CONNAME(192.0.2.1)        CURRENT
CHLTYPE(CLUSSDR)         STATUS(RUNNING)
RQMNAME(LONDON)
```

Send messages between the two queue managers, using **amqsput**.

3. On LONDON run the command **amqsput INVENTQ LONDON**.

   Type some messages, followed by a blank line.
4. On NEWYORK run the command **amqsget INVENTQ NEWYORK**.

   You now see the messages you entered on LONDON. After 15 seconds the program ends.

Send messages between the two queue managers using your own programs.

In the following steps, LONDON puts a message to the INVENTQ at NEWYORK and receives a reply on its queue LONDON_reply.

5. On LONDON put a messages to the cluster queue.
   a) Define a local queue called LONDON_reply.
   b) Set the MQOPEN options to MQOO_OUTPUT.
   c) Issue the MQOPEN call to open the queue INVENTQ.
   d) Set the *ReplyToQ* name in the message descriptor to LONDON_reply.
   e) Issue the MQPUT call to put the message.
   f) Commit the message.
6. On NEWYORK receive the message on the cluster queue and put a reply to the reply queue.
   a) Set the MQOPEN options to MQOO_BROWSE.
   b) Issue the MQOPEN call to open the queue INVENTQ.
   c) Issue the MQGET call to get the message from INVENTQ.
   d) Retrieve the *ReplyToQ* name from the message descriptor.
   e) Put the *ReplyToQ* name in the ObjectName field of the object descriptor.
   f) Set the MQOPEN options to MQOO_OUTPUT.
   g) Issue the MQOPEN call to open LONDON_reply at queue manager LONDON.
   h) Issue the MQPUT call to put the message to LONDON_reply.
7. On LONDON receive the reply.
   a) Set the MQOPEN options to MQOO_BROWSE.
   b) Issue the MQOPEN call to open the queue LONDON_reply.
   c) Issue the MQGET call to get the message from LONDON_reply.

Adding a queue manager to a cluster

Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using the single cluster transmission queue SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The INVENTORY cluster is set up as described in "Setting up a new cluster" on page 227. It contains two queue managers, LONDON and NEWYORK, which both hold full repositories.
- The queue manager PARIS is owned by the primary installation. If it is not, you must run the **setmqenv** command to set up the command environment for the installation that PARIS belongs to.
- TCP connectivity exists between all three systems, and the queue manager is configured with a TCP listener that starts under the control of the queue manager.

About this task

1. A new branch of the chain store is being set up in Paris and you want to add a queue manager called PARIS to the cluster.
2. Queue manager PARIS sends inventory updates to the application running on the system in New York by putting messages on the INVENTQ queue.

Follow these steps to add a queue manager to a cluster.

Procedure

1. Decide which full repository PARIS refers to first.

   Every queue manager in a cluster must refer to one or other of the full repositories. It gathers information about the cluster from a full repository and so builds up its own partial repository. Choose either of the repositories as the full repository. As soon as a new queue manager is added to the cluster it immediately learns about the other repository as well. Information about changes to a queue manager is sent directly to two repositories. In this example, you link PARIS to the queue manager LONDON, purely for geographical reasons.

   Note: Perform the remaining steps in any order, after queue manager PARIS is started.
2. Define a CLUSRCVR channel on queue manager PARIS.

   Every queue manager in a cluster must define a cluster-receiver channel on which it can receive messages. On PARIS, define:

   ```
   DEFINE CHANNEL(INVENTORY.PARIS) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(PARIS.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('Cluster-receiver channel for queue manager PARIS')
   ```

   The cluster-receiver channel advertises the availability of the queue manager to receive messages from other queue managers in the cluster INVENTORY. Do not make definitions on other queue managers for a sending end to the cluster-receiver channel INVENTORY.PARIS. Other definitions are made automatically when needed. See Cluster channels.
3. z/OS

   Start the channel initiator on IBM MQ for z/OS.
4. Define a CLUSSDR channel on queue manager PARIS.

   When you add to a cluster a queue manager that is not a full repository, you define just one cluster-sender channel to make an initial connection to a full repository. See Cluster-sender channel: CLUSSDR.

   On PARIS, make the following definition for a CLUSSDR channel called INVENTORY.LONDON to the queue manager with the network address LONDON.CHSTORE.COM.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('Cluster-sender channel from PARIS to repository at LONDON')
   ```

5. Optional: If you are adding to a cluster a queue manager that has previously been removed from the same cluster, check that it is now showing as a cluster member. If not, complete the following extra steps:
   a) Issue the **REFRESH CLUSTER** command on the queue manager you are adding.

      This step stops the cluster channels, and gives your local cluster cache a fresh set of sequence numbers that are assured to be up-to-date within the rest of the cluster.

      ```
      REFRESH CLUSTER(INVENTORY) REPOS(YES)
      ```

      Note: For large clusters, using the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.
   b) Restart the CLUSSDR channel

      (for example, using the START CHANNEL command).
   c) Restart the CLUSRCVR channel.

Results

The following figure shows the cluster set up by this task.

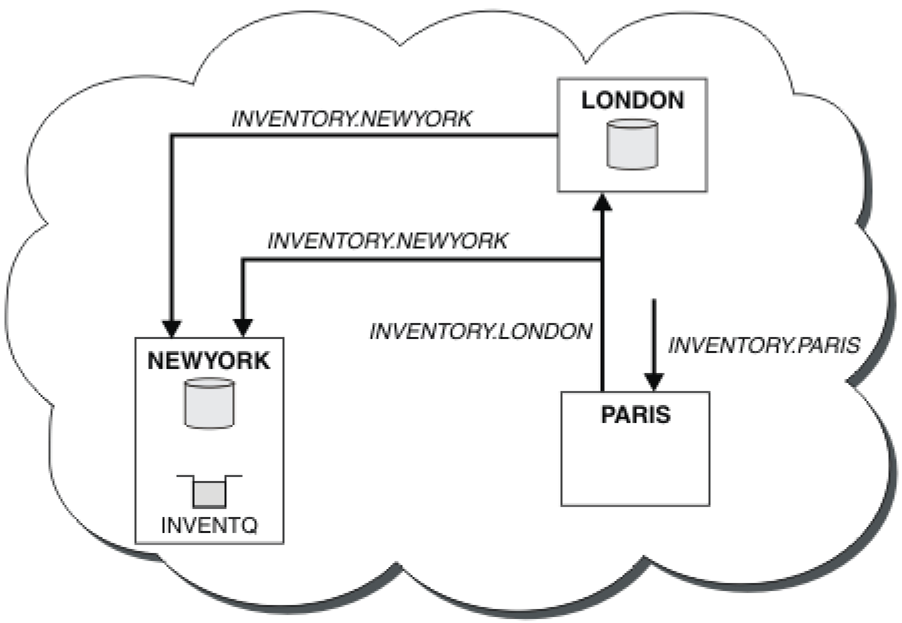

Figure 40. The INVENTORY cluster with three queue managers

By making only two definitions, a CLUSRCVR definition and a CLUSSDR definition, we added the queue manager PARIS to the cluster.

Now the PARIS queue manager learns, from the full repository at LONDON, that the INVENTQ queue is hosted by queue manager NEWYORK. When an application hosted by the system in Paris tries to put messages to the INVENTQ, PARIS automatically defines a cluster-sender channel to connect to the cluster-receiver channel INVENTORY.NEWYORK. The application can receive responses when its queue-manager name is specified as the target queue manager and a reply-to queue is provided.

Adding a queue manager to a cluster: separate transmission queues

Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using multiple cluster transmission queues.

Before you begin

- The queue manager is not a member of any clusters.
- The cluster exists; there is a full repository to which this queue manager can connect directly and the repository is available. For the steps to create the cluster, see "Setting up a new cluster" on page 227.

About this task

This task is an alternative to "Adding a queue manager to a cluster" on page 237, in which you add a queue manager to a cluster that places cluster messages on a single transmission queue.

In this task, you add a queue manager to a cluster that automatically creates separate cluster transmission queues for each cluster-sender channel.

To keep the number of definitions of queues small, the default is to use a single transmission queue. Using separate transmission queues is advantageous if you want to monitor traffic destined to different queue managers and different clusters. You might also want to separate traffic to different destinations to achieve isolation or performance goals.

Procedure

1. Alter the default cluster channel transmission queue type.

   Alter the queue manager PARIS:

   ```
   ALTER QMGR DEFCLXQ(CHANNEL)
   ```

   Every time the queue manager creates a cluster-sender channel to send a message to a queue manager, it creates a cluster transmission queue. The transmission queue is used only by this cluster-sender channel. The transmission queue is permanent-dynamic. It is created from the model queue, `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`, with the name `SYSTEM.CLUSTER.TRANSMIT.`*`ChannelName`*.

   Attention: If you are using dedicated `SYSTEM.CLUSTER.TRANSMIT.QUEUES` with a queue manager that was upgraded from a version of the product earlier than IBM WebSphere MQ 7.5, ensure that the `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE` has the SHARE/NOSHARE option set to **SHARE**.
2. Decide which full repository PARIS refers to first.

   Every queue manager in a cluster must refer to one or other of the full repositories. It gathers information about the cluster from a full repository and so builds up its own partial repository. Choose either of the repositories as the full repository. As soon as a new queue manager is added to the cluster it immediately learns about the other repository as well. Information about changes to a queue manager is sent directly to two repositories. In this example, you link PARIS to the queue manager `LONDON`, purely for geographical reasons.

   Note: Perform the remaining steps in any order, after queue manager PARIS is started.
3. Define a `CLUSRCVR` channel on queue manager PARIS.

   Every queue manager in a cluster must define a cluster-receiver channel on which it can receive messages. On PARIS, define:

   ```
   DEFINE CHANNEL(INVENTORY.PARIS) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(PARIS.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('Cluster-receiver channel for queue manager PARIS')
   ```

The cluster-receiver channel advertises the availability of the queue manager to receive messages from other queue managers in the cluster INVENTORY. Do not make definitions on other queue managers for a sending end to the cluster-receiver channel INVENTORY.PARIS. Other definitions are made automatically when needed. See Cluster channels.

4. Define a CLUSSDR channel on queue manager PARIS.

   When you add to a cluster a queue manager that is not a full repository, you define just one cluster-sender channel to make an initial connection to a full repository. See Cluster-sender channel: CLUSSDR .

   On PARIS, make the following definition for a CLUSSDR channel called INVENTORY.LONDON to the queue manager with the network address LONDON.CHSTORE.COM.

   ```
   DEFINE CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('Cluster-sender channel from PARIS to repository at LONDON')
   ```

   The queue manager automatically creates the permanent dynamic cluster transmission queue SYSTEM.CLUSTER.TRANSMIT.INVENTORY.LONDON from the model queue SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. It sets the CLCHNAME attribute of the transmission queue to INVENTORY.LONDON.

Results

The following figure shows the cluster set up by this task.

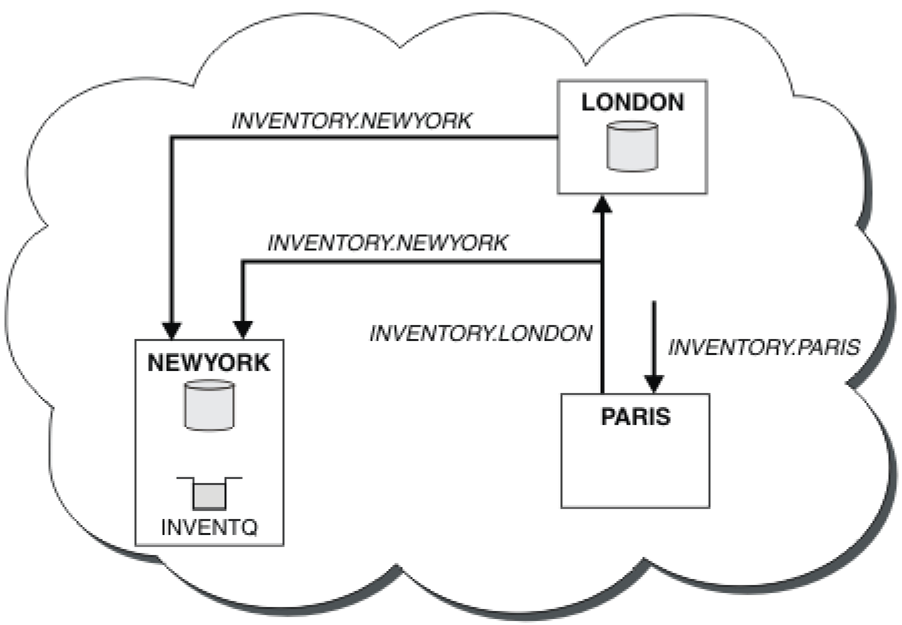

Figure 41. The INVENTORY cluster with three queue managers

By making only two definitions, a CLUSRCVR definition and a CLUSSDR definition, we added the queue manager PARIS to the cluster.

Now the PARIS queue manager learns, from the full repository at LONDON, that the INVENTQ queue is hosted by queue manager NEWYORK. When an application hosted by the system in Paris tries to put messages to the INVENTQ, PARIS automatically defines a cluster-sender channel to connect to the cluster-receiver channel INVENTORY.NEWYORK. The application can receive responses when its queue-manager name is specified as the target queue manager and a reply-to queue is provided.

Related tasks

Adding a queue manager to a cluster by using DHCP

Add a queue manager to a cluster, using DHCP. The task demonstrates omitting CONNAME value on a CLUSRCVR definition.

Adding a queue manager to a cluster by using DHCP

Add a queue manager to a cluster, using DHCP. The task demonstrates omitting CONNAME value on a CLUSRCVR definition.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

The task demonstrates two special features:

- The ability to omit the CONNAME value on a CLUSRCVR definition.
- The ability to use +QMNAME+ on a CLUSSDR definition.

Neither feature is provided on z/OS.

Scenario:

- The INVENTORY cluster has been set up as described in "Setting up a new cluster" on page 227. It contains two queue managers, LONDON and NEWYORK, which both hold full repositories.
- A new branch of the chain store is being set up in Paris and you want to add a queue manager called PARIS to the cluster.
- Queue manager PARIS sends inventory updates to the application running on the system in New York by putting messages on the INVENTQ queue.
- Network connectivity exists between all three systems.
- The network protocol is TCP.
- The PARIS queue manager system uses DHCP, which means that the IP addresses might change on system restart.
- The channels between the PARIS and LONDON systems are named according to a defined naming convention. The convention uses the queue manager name of the full repository queue manager on LONDON.
- Administrators of the PARIS queue manager have no information about the name of the queue manager on the LONDON repository. The name of the queue manager on the LONDON repository is subject to change.

About this task

Follow these steps to add a queue manager to a cluster by using DHCP.

Procedure

1. Decide which full repository PARIS refers to first.

   Every queue manager in a cluster must refer to one or other of the full repositories. It gathers information about the cluster from a full repository and so builds up its own partial repository. Choose either of the repositories as the full repository. As soon as a new queue manager is added to the cluster it immediately learns about the other repository as well. Information about changes to a queue manager is sent directly to two repositories. In this example we choose to link PARIS to the queue manager LONDON, purely for geographical reasons.

   Note: Perform the remaining steps in any order, after queue manager PARIS is started.
2. Define a CLUSRCVR channel on queue manager PARIS.

Every queue manager in a cluster needs to define a cluster-receiver channel on which it can receive messages. On PARIS, define:

```
DEFINE CHANNEL(INVENTORY.PARIS) CHLTYPE(CLUSRCVR)
TRPTYPE(TCP) CLUSTER(INVENTORY)
DESCR('Cluster-receiver channel for queue manager PARIS')
```

The cluster-receiver channel advertises the availability of the queue manager to receive messages from other queue managers in the cluster INVENTORY. You do not need to specify the CONNAME on the cluster-receiver channel. You can request IBM MQ to find out the connection name from the system, either by omitting CONNAME, or by specifying CONNAME(' '). IBM MQ generates the CONNAME value using the current IP address of the system; see CONNAME . There is no need to make definitions on other queue managers for a sending end to the cluster-receiver channel INVENTORY.PARIS. Other definitions are made automatically when needed.

3. Define a CLUSSDR channel on queue manager PARIS.

   Every queue manager in a cluster needs to define one cluster-sender channel on which it can send messages to its initial full repository. On PARIS, make the following definition for a channel called INVENTORY.+QMNAME+ to the queue manager with the network address LONDON.CHSTORE.COM.

   ```
   DEFINE CHANNEL(INVENTORY.+QMNAME+) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(LONDON.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('Cluster-sender channel from PARIS to repository at LONDON')
   ```

4. Optional: If you are adding to a cluster a queue manager that has previously been removed from the same cluster, check that it is now showing as a cluster member. If not, complete the following extra steps:

   a) Issue the **REFRESH CLUSTER** command on the queue manager you are adding.

      This step stops the cluster channels, and gives your local cluster cache a fresh set of sequence numbers that are assured to be up-to-date within the rest of the cluster.

      ```
      REFRESH CLUSTER(INVENTORY) REPOS(YES)
      ```

      Note: For large clusters, using the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

   b) Restart the CLUSSDR channel

      (for example, using the START CHANNEL command).

   c) Restart the CLUSRCVR channel.

Results

The cluster set up by this task is the same as for “Adding a queue manager to a cluster” on page 237:

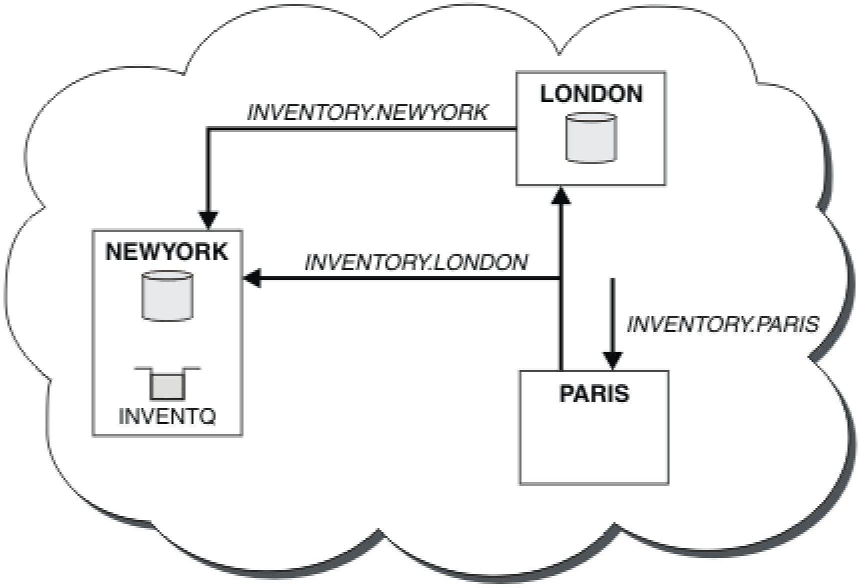

Figure 42. The INVENTORY cluster with three queue managers

By making only two definitions, a CLUSRCVR definition and a CLUSSDR definition, we have added the queue manager PARIS to the cluster.

On the PARIS queue manager, the CLUSSDR containing the string +QMNAME+ starts. On the LONDON system IBM MQ resolves the +QMNAME+ to the queue manager name (LONDON). IBM MQ then matches the definition for a channel called INVENTORY.LONDON to the corresponding CLUSRCVR definition.

IBM MQ sends back the resolved channel name to the PARIS queue manager. At PARIS, the CLUSSDR channel definition for the channel called INVENTORY.+QMNAME+ is replaced by an internally generated CLUSSDR definition for INVENTORY.LONDON. This definition contains the resolved channel name, but otherwise is the same as the +QMNAME+ definition that you made. The cluster repositories are also brought up to date with the channel definition with the newly resolved channel name.

Note:

1. The channel created with the +QMNAME+ name becomes inactive immediately. It is never used to transmit data.
2. Channel exits might see the channel name change between one invocation and the next.

Now the PARIS queue manager learns, from the repository at LONDON, that the INVENTQ queue is hosted by queue manager NEWYORK. When an application hosted by the system in Paris tries to put messages to the INVENTQ, PARIS automatically defines a cluster-sender channel to connect to the cluster-receiver channel INVENTORY.NEWYORK. The application can receive responses when its queue-manager name is specified as the target queue manager and a reply-to queue is provided.

Related tasks

Adding a queue manager to a cluster: separate transmission queues
Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using multiple cluster transmission queues.

Related reference

DEFINE CHANNEL

Adding a queue manager that hosts a queue

Add another queue manager to the cluster, to host another INVENTQ queue. Requests are sent alternately to the queues on each queue manager. No changes need to be made to the existing INVENTQ host.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The INVENTORY cluster has been set up as described in "Adding a queue manager to a cluster" on page 237. It contains three queue managers; LONDON and NEWYORK both hold full repositories, PARIS holds a partial repository. The inventory application runs on the system in New York, connected to the NEWYORK queue manager. The application is driven by the arrival of messages on the INVENTQ queue.
- A new store is being set up in Toronto. To provide additional capacity you want to run the inventory application on the system in Toronto as well as New York.
- Network connectivity exists between all four systems.
- The network protocol is TCP.

Note: The queue manager TORONTO contains only a partial repository. If you want to add a full-repository queue manager to a cluster, refer to "Moving a full repository to another queue manager" on page 249.

About this task

Follow these steps to add a queue manager that hosts a queue.

Procedure

1. Decide which full repository TORONTO refers to first.

   Every queue manager in a cluster must refer to one or other of the full repositories. It gathers information about the cluster from a full repository and so builds up its own partial repository. It is of no particular significance which repository you choose. In this example, we choose NEWYORK. Once the new queue manager has joined the cluster it communicates with both of the repositories.
2. Define the CLUSRCVR channel.

   Every queue manager in a cluster needs to define a cluster-receiver channel on which it can receive messages. On TORONTO, define a CLUSRCVR channel:

   ```
   DEFINE CHANNEL(INVENTORY.TORONTO) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(TORONTO.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('Cluster-receiver channel for TORONTO')
   ```

   The TORONTO queue manager advertises its availability to receive messages from other queue managers in the INVENTORY cluster using its cluster-receiver channel.
3. Define a CLUSSDR channel on queue manager TORONTO.

   Every queue manager in a cluster needs to define one cluster-sender channel on which it can send messages to its first full repository. In this case choose NEWYORK. TORONTO needs the following definition:

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('Cluster-sender channel from TORONTO to repository at NEWYORK')
   ```
4. Optional: If you are adding to a cluster a queue manager that has previously been removed from the same cluster, check that it is now showing as a cluster member. If not, complete the following extra steps:

a) Issue the **REFRESH CLUSTER** command on the queue manager you are adding.

This step stops the cluster channels, and gives your local cluster cache a fresh set of sequence numbers that are assured to be up-to-date within the rest of the cluster.

```
REFRESH CLUSTER(INVENTORY) REPOS(YES)
```

Note: For large clusters, using the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

b) Restart the CLUSSDR channel

(for example, using the START CHANNEL command).

c) Restart the CLUSRCVR channel.

5. Review the inventory application for message affinities.

   Before proceeding, ensure that the inventory application does not have any dependencies on the sequence of processing of messages and install the application on the system in Toronto.

6. Define the cluster queue `INVENTQ`.

   The `INVENTQ` queue, which is already hosted by the `NEWYORK` queue manager, is also to be hosted by `TORONTO`. Define it on the `TORONTO` queue manager as follows:

```
DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
```

Results

Figure 43 on page 246 shows the `INVENTORY` cluster set up by this task.

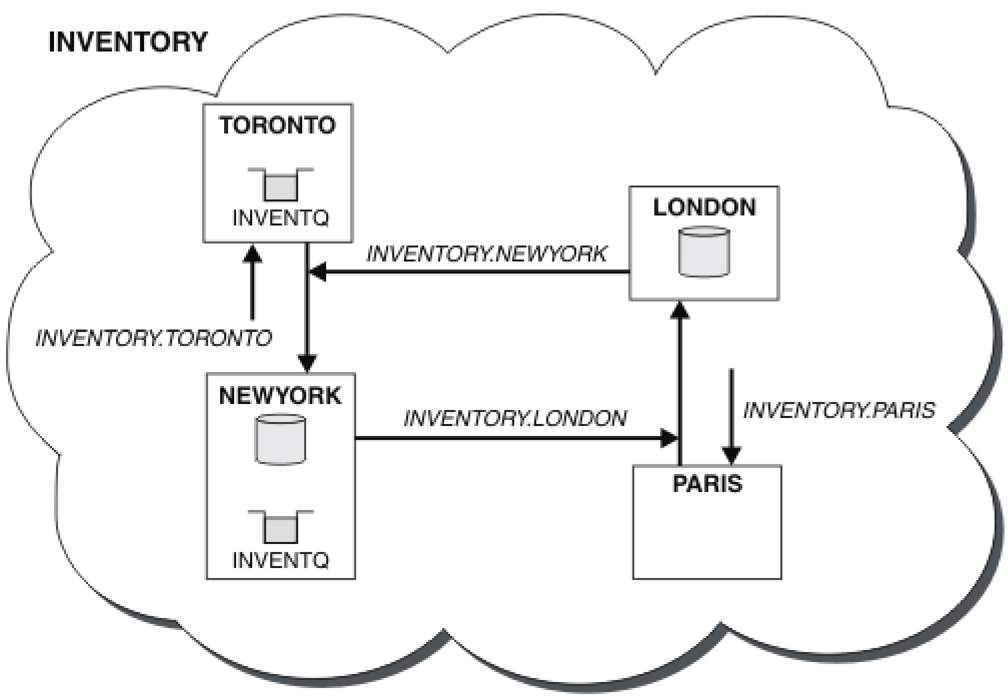

Figure 43. The `INVENTORY` cluster with four queue managers

The `INVENTQ` queue and the inventory application are now hosted on two queue managers in the cluster. This increases their availability, speeds up throughput of messages, and allows the workload to be distributed between the two queue managers. Messages put to `INVENTQ` by either `TORONTO` or `NEWYORK` are handled by the instance on the local queue manager whenever possible. Messages put by LONDON or PARIS are routed alternately to TORONTO or NEWYORK, so that the workload is balanced.

This modification to the cluster was accomplished without you having to alter the definitions on queue managers NEWYORK, LONDON, and PARIS. The full repositories in these queue managers are updated automatically with the information they need to be able to send messages to INVENTQ at TORONTO. The inventory application continues to function if one of the NEWYORK or the TORONTO queue manager becomes unavailable, and it has sufficient capacity. The inventory application must be able to work correctly if it is hosted in both locations.

As you can see from the result of this task, you can have the same application running on more than one queue manager. You can clustering to distribution workload evenly.

An application might not be able to process records in both locations. For example, suppose that you decide to add a customer-account query and update application running in LONDON and NEWYORK. An account record can only be held in one place. You could decide to control the distribution of requests by using a data partitioning technique. You can split the distribution of the records. You could arrange for half the records, for example for account numbers 00000 - 49999, to be held in LONDON. The other half, in the range 50000 - 99999 , are held in NEWYORK. You could then write a cluster workload exit program to examine the account field in all messages, and route the messages to the appropriate queue manager.

What to do next

Now that you have completed all the definitions, if you have not already done so start the channel initiator on IBM MQ for z/OS. On all platforms, start a listener program on queue manager TORONTO. The listener program waits for incoming network requests and starts the cluster-receiver channel when it is needed.

Adding a queue-sharing group to existing clusters

Add a queue-sharing group on z/OS to existing clusters.

Before you begin

Note:

1. For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.
2. Queue-sharing groups are supported only on IBM MQ for z/OS. This task is not applicable to other platforms.

Scenario:

- The INVENTORY cluster has been set up as described in "Setting up a new cluster" on page 227. It contains two queue managers, LONDON and NEWYORK.
- You want to add a queue-sharing group to this cluster. The group, QSGP, comprises three queue managers, P1, P2, and P3. They share an instance of the INVENTQ queue, which is to be defined by P1.

About this task

Follow these steps to add new queue managers that host a shared queue.

Procedure

1. Decide which full repository the queue managers refer to first.

   Every queue manager in a cluster must refer to one or other of the full repositories. It gathers information about the cluster from a full repository and so builds up its own partial repository. It is of no particular significance which full repository you choose. In this example, choose NEWYORK. Once the queue-sharing group has joined the cluster, it communicates with both of the full repositories.
2. Define the CLUSRCVR channels.

Every queue manager in a cluster needs to define a cluster-receiver channel on which it can receive messages. On P1, P2, and P3, define:

```
DEFINE CHANNEL(INVENTORY.Pn) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
CONNAME(Pn.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('Cluster-receiver channel for sharing queue manager')
```

The cluster-receiver channel advertises the availability of each queue manager to receive messages from other queue managers in the cluster `INVENTORY`.

3. Define a `CLUSSDR` channel for the queue-sharing group.

   Every member of a cluster needs to define one cluster-sender channel on which it can send messages to its first full repository. In this case we have chosen `NEWYORK`. One of the queue managers in the queue-sharing group needs the following group definition. The definition ensures that every queue manager has a cluster-sender channel definition.

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY) QSGDISP(GROUP)
   DESCR('Cluster-sender channel to repository at NEWYORK')
   ```

4. Define the shared queue.

   Define the queue `INVENTQ` on P1 as follows:

   ```
   DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY) QSGDISP(SHARED) CFSTRUCT(STRUCTURE)
   ```

   Start the channel initiator and a listener program on the new queue manager. The listener program listens for incoming network requests and starts the cluster-receiver channel when it is needed.

Results

Figure 44 on page 248 shows the cluster set up by this task.

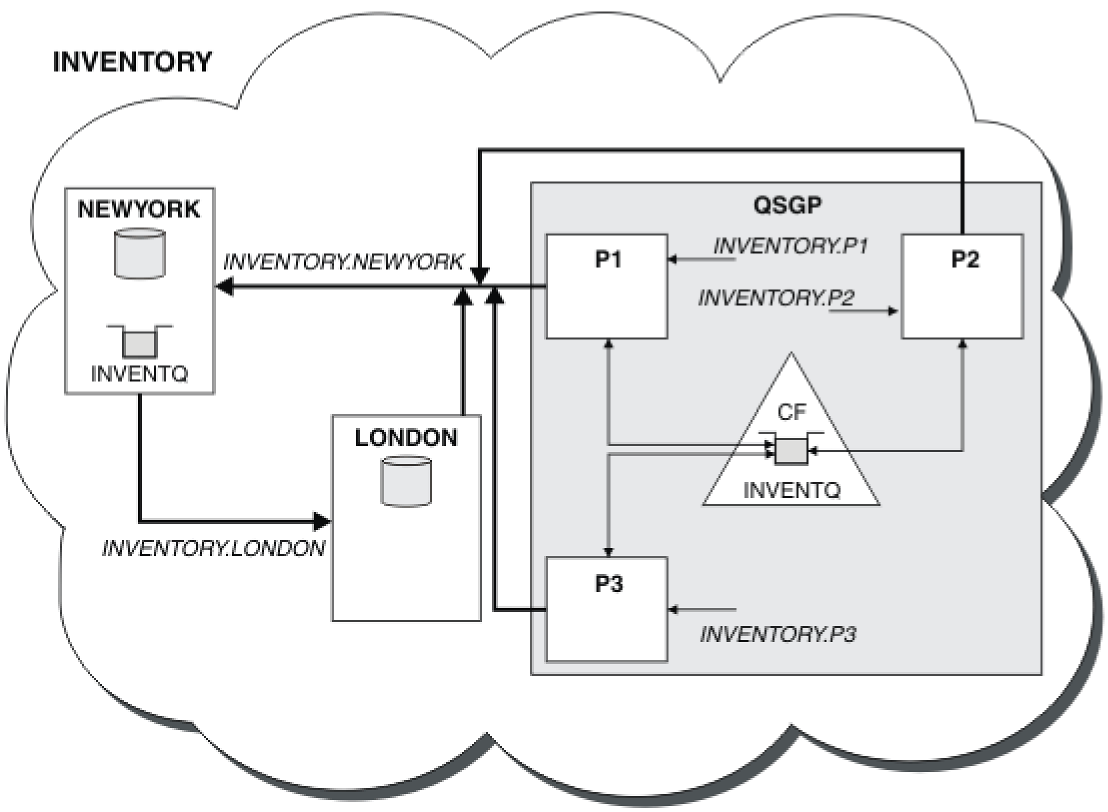

Figure 44. Cluster and queue-sharing group

Now messages put on the INVENTQ queue by LONDON are routed alternately around the four queue managers advertised as hosting the queue.

What to do next

A benefit of having members of a queue-sharing group host a cluster queue is any member of the group can reply to a request. In this case perhaps P1 becomes unavailable after receiving a message on the shared queue. Another member of the queue-sharing group can reply instead.

Moving a full repository to another queue manager

Move a full repository from one queue manager to another, building up the new repository from information held at the second repository.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The INVENTORY cluster has been set up as described in "Adding a queue manager to a cluster" on page 237.
- For business reasons you now want to remove the full repository from queue manager LONDON, and replace it with a full repository at queue manager PARIS. The NEWYORK queue manager is to continue holding a full repository.

About this task

Follow these steps to move a full repository to another queue manager.

Procedure

1. Alter PARIS to make it a full repository queue manager.

   On PARIS, issue the following command:

   ```
   ALTER QMGR REPOS(INVENTORY)
   ```

2. Add a CLUSSDR channel on PARIS

   PARIS currently has a cluster-sender channel pointing to LONDON. LONDON is no longer to hold a full repository for the cluster. PARIS must have a new cluster-sender channel that points to NEWYORK, where the other full repository is now held.

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('Cluster-sender channel from PARIS to repository at NEWYORK')
   ```

3. Define a CLUSSDR channel on NEWYORK that points to PARIS

   Currently NEWYORK has a cluster-sender channel pointing to LONDON. Now that the other full repository has moved to PARIS, you need to add a new cluster-sender channel at NEWYORK that points to PARIS.

   ```
   DEFINE CHANNEL(INVENTORY.PARIS) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(PARIS.CHSTORE.COM) CLUSTER(INVENTORY)
   DESCR('Cluster-sender channel from NEWYORK to repository at PARIS')
   ```

   When you add the cluster-sender channel to PARIS, PARIS learns about the cluster from NEWYORK. It builds up its own full repository using the information from NEWYORK.

4. Check that queue manager PARIS now has a full repository

   Check that queue manager PARIS has built its own full repository from the full repository on queue manager NEWYORK. Issue the following commands:

   ```
   DIS QCLUSTER(*) CLUSTER (INVENTORY)
   DIS CLUSQMGR(*) CLUSTER (INVENTORY)
   ```

   Check that these commands show details of the same resources in this cluster as on NEWYORK.

   Note: If queue manager NEWYORK is not available, this building of information cannot complete. Do not move on to the next step until the task is complete.
5. Alter the queue-manager definition on LONDON

   Finally alter the queue manager at LONDON so that it no longer holds a full repository for the cluster. On LONDON, issue the command:

   ```
   ALTER QMGR REPOS(' ')
   ```

   The queue manager no longer receives any cluster information. After 30 days the information that is stored in its full repository expires. The queue manager LONDON now builds up its own partial repository.
6. Remove or change any outstanding definitions.

   When you are sure that the new arrangement of your cluster is working as expected, remove or change manually defined CLUSSDR definitions that are no longer correct.

   - On the PARIS queue manager, you must stop and delete the cluster-sender channel to LONDON, and then issue the start channel command so that the cluster can use the automatic channels again:

     ```
     STOP CHANNEL(INVENTORY.LONDON)
     DELETE CHANNEL(INVENTORY.LONDON)
     START CHANNEL(INVENTORY.LONDON)
     ```

   - On the NEWYORK queue manager, you must stop and delete the cluster-sender channel to LONDON, and then issue the start channel command so that the cluster can use the automatic channels again:

     ```
     STOP CHANNEL(INVENTORY.LONDON)
     DELETE CHANNEL(INVENTORY.LONDON)
     START CHANNEL(INVENTORY.LONDON)
     ```

   - Replace all other manually defined cluster-sender channels that point to LONDON on all queue managers in the cluster with channels that point to either NEWYORK or PARIS. After deleting a channel, always issue the **start channel** command so that the cluster can use the automatic channels again. In this small example, there are no others. To check whether there are any others that you have forgotten, issue the DISPLAY CHANNEL command from each queue manager, specifying TYPE(CLUSSDR). For example:

     ```
     DISPLAY CHANNEL(*) TYPE(CLUSSDR)
     ```

   It is important that you perform this task as soon as possible after moving the full repository from LONDON to PARIS. In the time before you perform this task, queue managers that have manually defined CLUSSDR channels named INVENTORY.LONDON might send requests for information using this channel.

   After LONDON has ceased to be a full repository, if it receives such requests it will write error messages to its queue manager error log. The following examples show which error messages might be seen on LONDON:

   - `AMQ9428: Unexpected publication of a cluster queue object received`

- `AMQ9432: Query received by a non-repository queue manager`

  The queue manager `LONDON` does not respond to the requests for information because it is no longer a full repository. The queue managers requesting information from `LONDON` must rely on `NEWYORK` for cluster information until their manually defined CLUSSDR definitions are corrected to point to `PARIS`. This situation must not be tolerated as a valid configuration in the long term.

Results

Figure 45 on page 251 shows the cluster set up by this task.

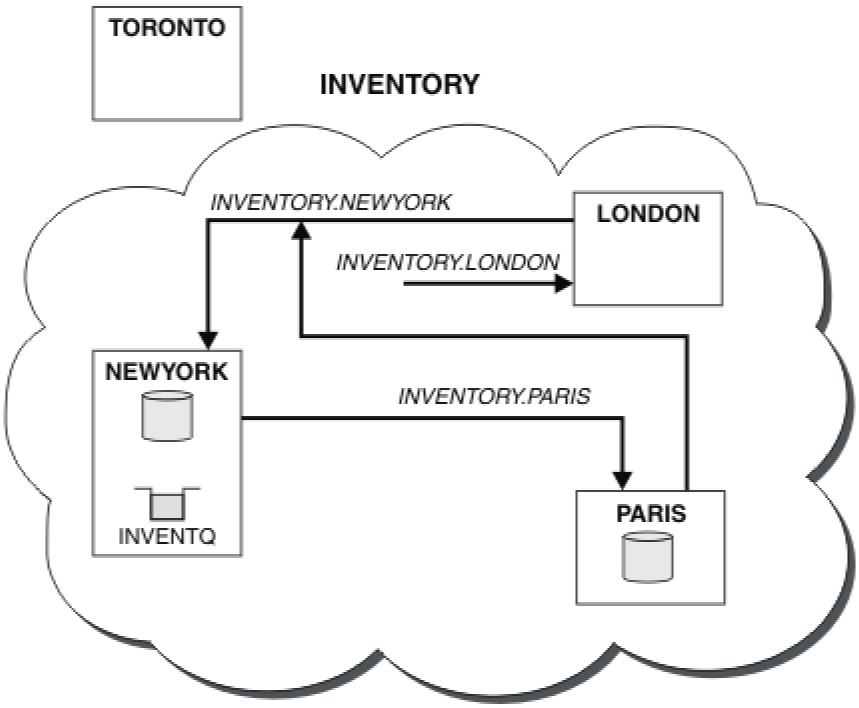

Figure 45. The INVENTORY cluster with the full repository moved to PARIS

Establishing communication in a cluster

A channel initiator is needed to start a communication channel when there is a message to deliver. A channel listener waits to start the other end of a channel to receive the message.

Before you begin

To establish communication between queue managers in a cluster, configure a link using one of the supported communication protocols. The supported protocols are TCP or LU 6.2 on any platform, and NetBIOS or SPX on Windows systems. As part of this configuration, you also need channel initiators and channel listeners just as you do with distributed queuing.

About this task

All cluster queue managers need a channel initiator to monitor the system-defined initiation queue `SYSTEM.CHANNEL.INITQ`. `SYSTEM.CHANNEL.INITQ` is the initiation queue for all transmission queues including the cluster transmission queue.

Each queue manager must have a channel listener. A channel listener program waits for incoming network requests and starts the appropriate receiver-channel when it is needed. The implementation of channel listeners is platform-specific, however there are some common features. On all IBM MQ platforms, the listener can be started using the `START LISTENER` command. On IBM MQ for IBM i, Windows, UNIX and

Linux systems, you can start the listener automatically at the same time as the queue manager. To start the listener automatically, set the CONTROL attribute of the LISTENER object to QMGR or STARTONLY.

z/OS A non-shared listener port (INDISP(QMGR)) must be used for CLUSRCVR channels on z/OS and for CLUSSDR channels to z/OS.

Procedure

1. Start the channel initiator.

   - z/OS

     IBM MQ for z/OS
     There is one channel initiator for each queue manager and it runs as a separate address space. You start it using the **MQSC** START CHINIT command, which you issue as part of your queue manager startup.

   - Windows Linux UNIX

     IBM MQ for Windows, UNIX and Linux systems
     When you start a queue manager, if the queue manager attribute SCHINIT is set to QMGR, a channel initiator is automatically started. Otherwise it can be started using the **runmqsc** START CHINIT command or the **runmqchi** control command.

   - IBM i

     IBM MQ for IBM i
     When you start a queue manager, if the queue manager attribute SCHINIT is set to QMGR, a channel initiator is automatically started. Otherwise it can be started using the **runmqsc** START CHINIT command or the **runmqchi** control command.

2. Start the channel listener.

   - z/OS

     IBM MQ for z/OS
     Use the channel listener program provided by IBM MQ. To start an IBM MQ channel listener, use the **MQSC** command START LISTENER, which you issue as part of your channel initiator startup. For example:

     ```
     START LISTENER PORT(1414) TRPTYPE(TCP)
     ```

     or:

     ```
     START LISTENER LUNAME(LONDON.LUNAME) TRPTYPE(LU62)
     ```

     Members of a queue-sharing group can use a shared listener instead of a listener for each queue manager. Do not use shared listeners with clusters. Specifically, do not make the CONNAME of the CLUSRCVR channel the address of the shared listener of the queue sharing group. If you do, queue managers might receive messages for queues for which they do not have a definition.

   - IBM i

     IBM MQ for IBM i
     Use the channel listener program provided by IBM MQ. To start an IBM MQ channel listener use the **CL** command STRMQMLSR. For example:

     ```
     STRMQMLSR MQMNAME(QM1) PORT(1414)
     ```

   - Windows

IBM MQ for Windows

Use either the channel listener program provided by IBM MQ, or the facilities provided by the operating system.

To start the IBM MQ channel listener use the RUNMQLSR command. For example:

```
RUNMQLSR -t tcp -p 1414 -m QM1
```

- Linux UNIX

IBM MQ on UNIX and Linux systems

Use either the channel listener program provided by IBM MQ, or the facilities provided by the operating system; for example, **inetd** for TCP communications.

To start the IBM MQ channel listener use the **runmqlsr** command. For example:

```
runmqlsr -t tcp -p 1414 -m QM1
```

To use **inetd** to start channels, configure two files:

a. Edit the file `/etc/services`. You must be logged in as a superuser or root. If the following line is not in the file, add it as shown:

```
MQSeries    1414/tcp   # Websphere MQ channel listener
```

where 1414 is the port number required by IBM MQ. You can change the port number, but it must match the port number specified at the sending end.

b. Edit the file `/etc/inetd.conf`. If you do not have the following line in that file, add it as shown:

```
MQSeries stream tcp nowait mqm MQ_INSTALLATION_PATH/bin/amqcrsta amqcrsta
-m queue.manager.name
```

where *MQ_INSTALLATION_PATH* is replaced by the high-level directory in which IBM MQ is installed.

The updates become active after **inetd** has reread the configuration files. Issue the following commands from the root user ID:

On AIX:

```
refresh -s inetd
```

On HP-UX:

```
inetd -c
```

On Solaris or Linux:

a. Find the process ID of the **inetd** with the command:

```
ps -ef | grep inetd
```

b. Run the appropriate command, as follows:

   – For Solaris 9 and Linux:

```
kill -1 inetd processid
```

– For Solaris 10, or later versions:

```
inetconv
```

Converting an existing network into a cluster

Convert an existed distributed queuing network to a cluster and add an additional queue manager to increase capacity.

Before you begin

In “Setting up a new cluster” on page 227 through “Moving a full repository to another queue manager” on page 249 you created and extended a new cluster. The next two tasks explore a different approach: that of converting an existing network of queue managers into a cluster.

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- A IBM MQ network is already in place, connecting the nationwide branches of a chain store. It has a hub and spoke structure: all the queue managers are connected to one central queue manager. The central queue manager is on the system on which the inventory application runs. The application is driven by the arrival of messages on the INVENTQ queue, for which each queue manager has a remote-queue definition.

  This network is illustrated in Figure 46 on page 254.

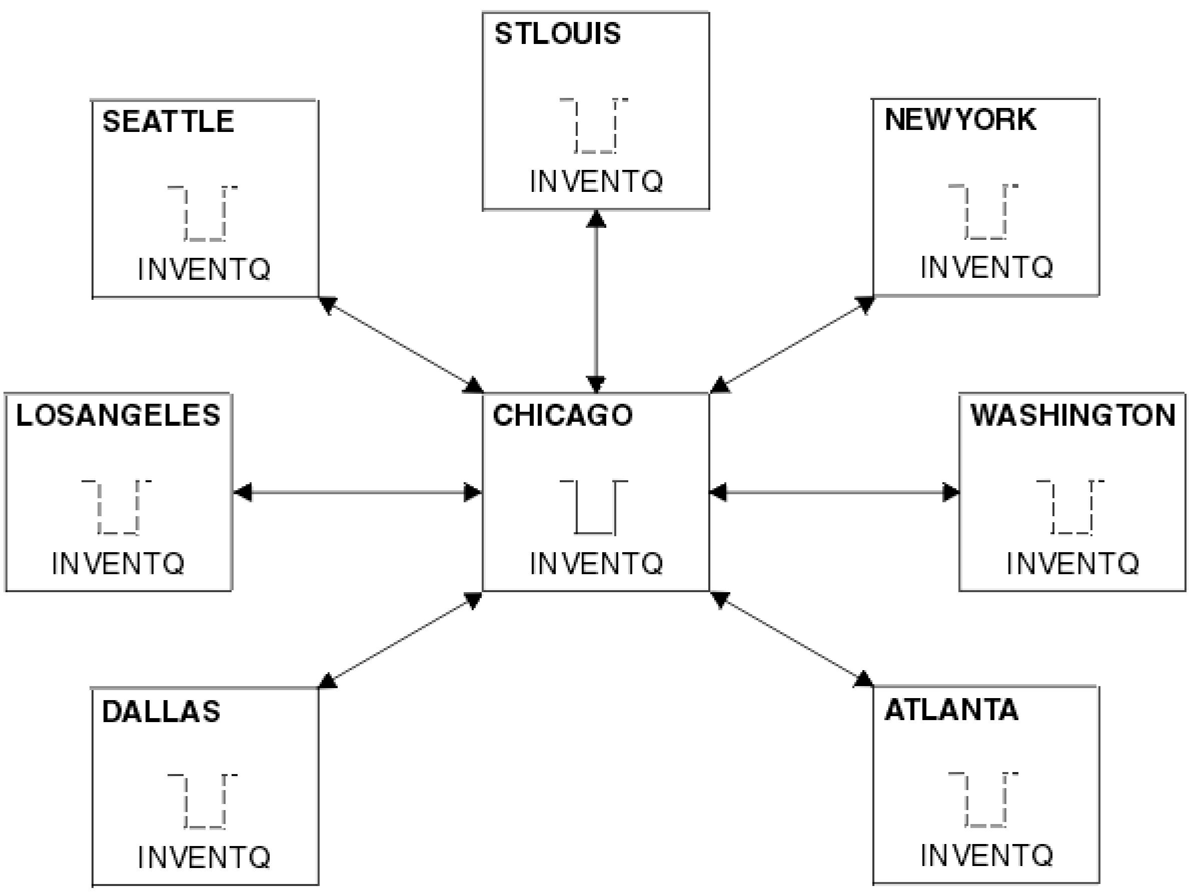

Figure 46. A hub and spoke network

- To ease administration you are going to convert this network into a cluster and create another queue manager at the central site to share the workload.

The cluster name is CHNSTORE.

Note: The cluster name CHNSTORE was selected to allow cluster-receiver channel names to be created using names in the format *cluster-name. queue-manager* that do not exceed the maximum length of 20 characters, for example CHNSTORE.WASHINGTON.

- Both the central queue managers are to host full repositories and be accessible to the inventory application.
- The inventory application is to be driven by the arrival of messages on the INVENTQ queue hosted by either of the central queue managers.
- The inventory application is to be the only application running in parallel and accessible by more than one queue manager. All other applications continue to run as before.
- All the branches have network connectivity to the two central queue managers.
- The network protocol is TCP.

About this task

Follow these steps to convert an existing network into a cluster.

Procedure

1. Review the inventory application for message affinities.

   Before proceeding ensure that the application can handle message affinities. Message affinities are the relationship between conversational messages that are exchanged between two applications, where the messages must be processed by a particular queue manager or in a particular sequence. For more information on message affinities, see: “Handling message affinities” on page 328
2. Alter the two central queue managers to make them full repository queue managers.

   The two queue managers CHICAGO and CHICAGO2 are at the hub of this network. You have decided to concentrate all activity associated with the chain store cluster on to those two queue managers. As well as the inventory application and the definitions for the INVENTQ queue, you want these queue managers to host the two full repositories for the cluster. At each of the two queue managers, issue the following command:

   ```
   ALTER QMGR REPOS(CHNSTORE)
   ```
3. Define a CLUSRCVR channel on each queue manager.

   At each queue manager in the cluster, define a cluster-receiver channel and a cluster-sender channel. It does not matter which channel you define first.

   Make a CLUSRCVR definition to advertise each queue manager, its network address, and other information, to the cluster. For example, on queue manager ATLANTA:

   ```
   DEFINE CHANNEL(CHNSTORE.ATLANTA) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(ATLANTA.CHSTORE.COM) CLUSTER(CHNSTORE)
   DESCR('Cluster-receiver channel')
   ```
4. Define a CLUSSDR channel on each queue manager

   Make a CLUSSDR definition at each queue manager to link that queue manager to one or other of the full repository queue managers. For example, you might link ATLANTA to CHICAGO2:

   ```
   DEFINE CHANNEL(CHNSTORE.CHICAGO2) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(CHICAGO2.CHSTORE.COM) CLUSTER(CHNSTORE)
   DESCR('Cluster-sender channel to repository queue manager')
   ```
5. Install the inventory application on CHICAGO2.

   You already have the inventory application on queue manager CHICAGO. Now you need to make a copy of this application on queue manager CHICAGO2.

6. Define the INVENTQ queue on the central queue managers.

   On CHICAGO, modify the local queue definition for the queue INVENTQ to make the queue available to the cluster. Issue the command:

   ```
   ALTER QLOCAL(INVENTQ) CLUSTER(CHNSTORE)
   ```

   On CHICAGO2, make a definition for the same queue:

   ```
   DEFINE QLOCAL(INVENTQ) CLUSTER(CHNSTORE)
   ```

   On z/OS, you can use the MAKEDEF option of the COMMAND function of **CSQUTIL** to make an exact copy on CHICAGO2 of the INVENTQ on CHICAGO.

   When you make these definitions, a message is sent to the full repositories at CHICAGO and CHICAGO2 and the information in them is updated. The queue manager finds out from the full repositories when it puts a message to the INVENTQ, that there is a choice of destinations for the messages.
7. Check that the cluster changes have been propagated.

   Check that the definitions you created in the previous step have been propagated though the cluster. Issue the following command on a full repository queue manager:

   ```
   DIS QCLUSTER(INVENTQ)
   ```

Adding a new, interconnected cluster

Add a new cluster that shares some queue managers with an existing cluster.

Before you begin

Note:

1. For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.
2. Before starting this task, check for queue-name clashes and understand the consequences. You might need to rename a queue, or set up queue aliases before you can proceed.

Scenario:

- An IBM MQ cluster has been set up as described in “Converting an existing network into a cluster” on page 254.
- A new cluster called MAILORDER is to be implemented. This cluster comprises four of the queue managers that are in the CHNSTORE cluster; CHICAGO, CHICAGO2, SEATTLE, and ATLANTA, and two additional queue managers; HARTFORD and OMAHA. The MAILORDER application runs on the system at Omaha, connected to queue manager OMAHA. It is driven by the other queue managers in the cluster putting messages on the MORDERQ queue.
- The full repositories for the MAILORDER cluster are maintained on the two queue managers CHICAGO and CHICAGO2.
- The network protocol is TCP.

About this task

Follow these steps to add a new, interconnected cluster.

Procedure

1. Create a namelist of the cluster names.

The full repository queue managers at CHICAGO and CHICAGO2 are now going to hold the full repositories for both of the clusters CHNSTORE and MAILORDER. First, create a namelist containing the names of the clusters. Define the namelist on CHICAGO and CHICAGO2, as follows:

```
DEFINE NAMELIST(CHAINMAIL)
DESCR('List of cluster names')
NAMES(CHNSTORE, MAILORDER)
```

2. Alter the two queue-manager definitions.

   Now alter the two queue-manager definitions at CHICAGO and CHICAGO2. Currently these definitions show that the queue managers hold full repositories for the cluster CHNSTORE. Change that definition to show that the queue managers hold full repositories for all clusters listed in the CHAINMAIL namelist. Alter the CHICAGO and CHICAGO2 queue manager definitions:

```
ALTER QMGR REPOS(' ') REPOSNL(CHAINMAIL)
```

3. Alter the CLUSRCVR channels on CHICAGO and CHICAGO2.

   The CLUSRCVR channel definitions at CHICAGO and CHICAGO2 show that the channels are available in the cluster CHNSTORE. You need to change the cluster-receiver definition to show that the channels are available to all clusters listed in the CHAINMAIL namelist. Change the cluster-receiver definition at CHICAGO:

```
ALTER CHANNEL(CHNSTORE.CHICAGO) CHLTYPE(CLUSRCVR)
CLUSTER(' ') CLUSNL(CHAINMAIL)
```

   At CHICAGO2, enter the command:

```
ALTER CHANNEL(CHNSTORE.CHICAGO2) CHLTYPE(CLUSRCVR)
CLUSTER(' ') CLUSNL(CHAINMAIL)
```

4. Alter the CLUSSDR channels on CHICAGO and CHICAGO2.

   Change the two CLUSSDR channel definitions to add the namelist. At CHICAGO, enter the command:

```
ALTER CHANNEL(CHNSTORE.CHICAGO2) CHLTYPE(CLUSSDR)
CLUSTER(' ') CLUSNL(CHAINMAIL)
```

   At CHICAGO2, enter the command:

```
ALTER CHANNEL(CHNSTORE.CHICAGO) CHLTYPE(CLUSSDR)
CLUSTER(' ') CLUSNL(CHAINMAIL)
```

5. Create a namelist on SEATTLE and ATLANTA.

   Because SEATTLE and ATLANTA are going to be members of more than one cluster, you must create a namelist containing the names of the clusters. Define the namelist on SEATTLE and ATLANTA, as follows:

```
DEFINE NAMELIST(CHAINMAIL)
DESCR('List of cluster names')
NAMES(CHNSTORE, MAILORDER)
```

6. Alter the CLUSRCVR channels on SEATTLE and ATLANTA.

The CLUSRCVR channel definitions at SEATTLE and ATLANTA show that the channels are available in the cluster CHNSTORE. Change the cluster-receive channel definitions to show that the channels are available to all clusters listed in the CHAINMAIL namelist. At SEATTLE, enter the command:

```
ALTER CHANNEL(CHNSTORE.SEATTLE) CHLTYPE(CLUSRCVR)
CLUSTER(' ') CLUSNL(CHAINMAIL)
```

At ATLANTA, enter the command:

```
ALTER CHANNEL(CHNSTORE.ATLANTA) CHLTYPE(CLUSRCVR)
CLUSTER(' ') CLUSNL(CHAINMAIL)
```

7. Alter the CLUSSDR channels on SEATTLE and ATLANTA.

   Change the two CLUSSDR channel definitions to add the namelist. At SEATTLE, enter the command:

```
ALTER CHANNEL(CHNSTORE.CHICAGO) CHLTYPE(CLUSSDR)
CLUSTER(' ') CLUSNL(CHAINMAIL)
```

   At ATLANTA, enter the command:

```
ALTER CHANNEL(CHNSTORE.CHICAGO2) CHLTYPE(CLUSSDR)
CLUSTER(' ') CLUSNL(CHAINMAIL)
```

8. Define CLUSRCVR and CLUSSDR channels on HARTFORD and OMAHA.

   On the two new queue managers HARTFORD and OMAHA, define cluster-receiver and cluster-sender channels. It does not matter in which sequence you make the definitions. At HARTFORD, enter:

```
DEFINE CHANNEL(MAILORDER.HARTFORD) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
CONNAME(HARTFORD.CHSTORE.COM) CLUSTER(MAILORDER)
DESCR('Cluster-receiver channel for HARTFORD')

DEFINE CHANNEL(MAILORDER.CHICAGO) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(CHICAGO.CHSTORE.COM) CLUSTER(MAILORDER)
DESCR('Cluster-sender channel from HARTFORD to repository at CHICAGO')
```

   At OMAHA, enter:

```
DEFINE CHANNEL(MAILORDER.OMAHA) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
CONNAME(OMAHA.CHSTORE.COM) CLUSTER(MAILORDER)
DESCR('Cluster-receiver channel for OMAHA')

DEFINE CHANNEL(MAILORDER.CHICAGO) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(CHICAGO.CHSTORE.COM) CLUSTER(MAILORDER)
DESCR('Cluster-sender channel from OMAHA to repository at CHICAGO')
```

9. Define the MORDERQ queue on OMAHA.

   The final step to complete this task is to define the queue MORDERQ on the queue manager OMAHA. At OMAHA, enter:

```
DEFINE QLOCAL(MORDERQ) CLUSTER(MAILORDER)
```

10. Check that the cluster changes have been propagated.

    Check that the definitions you created with the previous steps have been propagated though the cluster. Issue the following commands on a full repository queue manager:

```
DIS QCLUSTER (MORDERQ)
DIS CLUSQMGR
```

11.

Results

The cluster set up by this task is shown in Figure 47 on page 260.

Now we have two overlapping clusters. The full repositories for both clusters are held at CHICAGO and CHICAGO2. The mail order application that runs on OMAHA is independent of the inventory application that runs at CHICAGO. However, some of the queue managers that are in the CHNSTORE cluster are also in the MAILORDER cluster, and so they can send messages to either application. Before carrying out this task to overlap two clusters, be aware of the possibility of queue-name clashes.

Suppose that on NEWYORK in cluster CHNSTORE and on OMAHA in cluster MAILORDER, there is a queue called ACCOUNTQ. If you overlap the clusters and then an application on SEATTLE puts a message to the queue ACCOUNTQ, the message can go to either instance of the ACCOUNTQ.

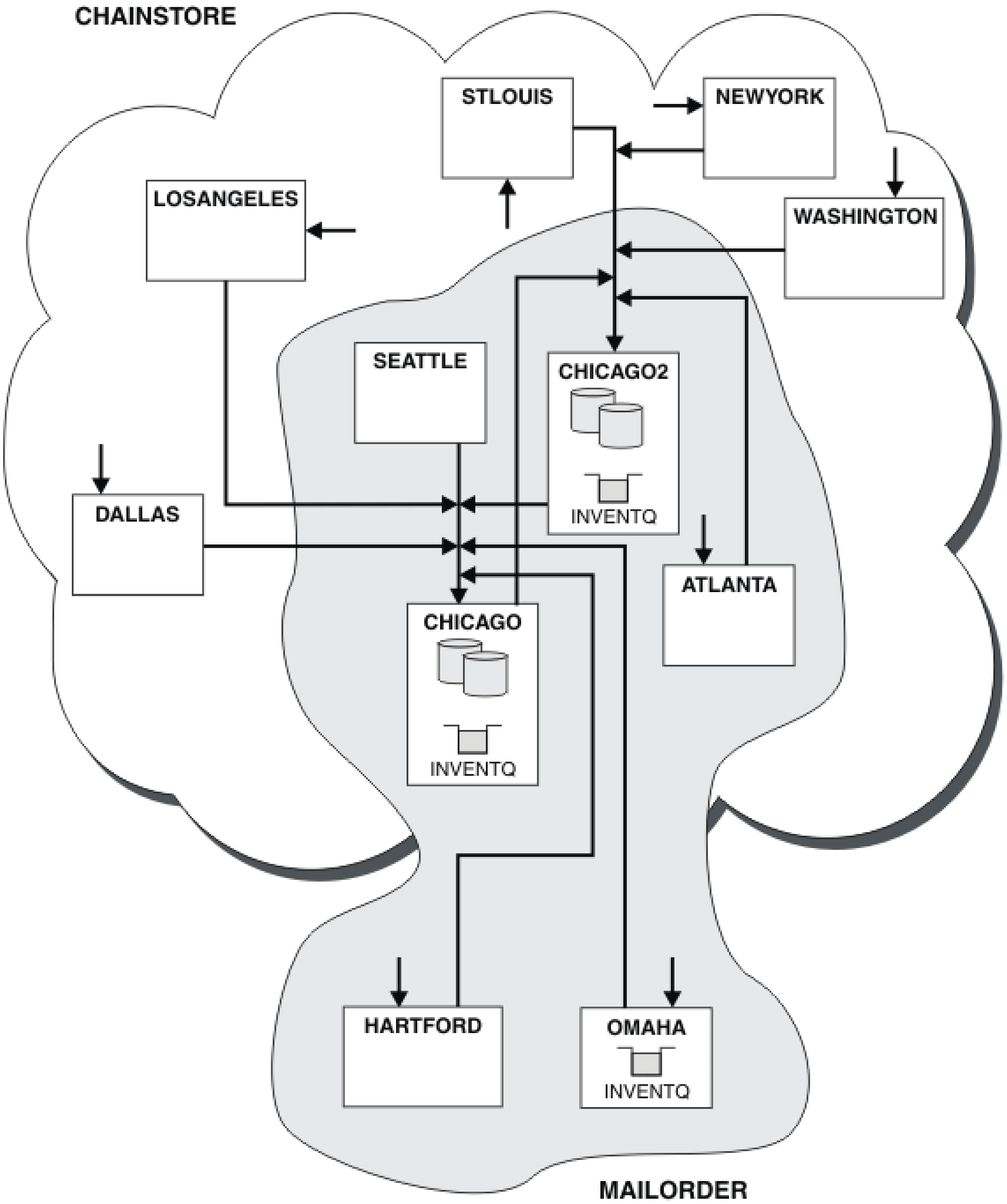

Figure 47. Interconnected clusters

What to do next

Suppose you decide to merge the MAILORDER cluster with the CHNSTORE cluster to form one large cluster called CHNSTORE.

To merge the MAILORDER cluster with the CHNSTORE cluster, such that CHICAGO and CHICAGO2 hold the full repositories:

- Alter the queue manager definitions for CHICAGO and CHICAGO2, removing the REPOSNL attribute, which specifies the namelist (CHAINMAIL), and replacing it with a REPOS attribute specifying the cluster name (CHNSTORE). For example:

```
ALTER QMGR(CHICAGO) REPOSNL(' ') REPOS(CHNSTORE)
```

- On each queue manager in the MAILORDER cluster, alter all the channel definitions and queue definitions to change the value of the CLUSTER attribute from MAILORDER to CHNSTORE. For example, at HARTFORD, enter:

```
ALTER CHANNEL(MAILORDER.HARTFORD) CLUSTER(CHNSTORE)
```

  At OMAHA enter:

```
ALTER QLOCAL(MORDERQ) CLUSTER(CHNSTORE)
```

- Alter all definitions that specify the cluster namelist CHAINMAIL, that is, the CLUSRCVR and CLUSSDR channel definitions at CHICAGO, CHICAGO2, SEATTLE, and ATLANTA, to specify instead the cluster CHNSTORE.

From this example, you can see the advantage of using namelists. Instead of altering the queue manager definitions for CHICAGO and CHICAGO2 you can alter the value of the namelist CHAINMAIL. Similarly, instead of altering the CLUSRCVR and CLUSSDR channel definitions at CHICAGO, CHICAGO2, SEATTLE, and ATLANTA, you can achieve the required result by altering the namelist.

Related tasks

Removing a cluster network
Remove a cluster from a network and restore the distributed queuing configuration.

Removing a cluster network

Remove a cluster from a network and restore the distributed queuing configuration.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- A IBM MQ cluster has been set up as described in "Converting an existing network into a cluster" on page 254.
- This cluster is now to be removed from the system. The network of queue managers is to continue functioning as it did before the cluster was implemented.

About this task

Follow these steps to remove a cluster network.

Procedure

1. Remove cluster queues from the CHNSTORE cluster.

   On both CHICAGO and CHICAGO2, modify the local queue definition for the queue INVENTQ to remove the queue from the cluster. Issue the command:

```
ALTER QLOCAL(INVENTQ) CLUSTER(' ')
```

   When you alter the queue, the information in the full repositories is updated and propagated throughout the cluster. Active applications using MQOO_BIND_NOT_FIXED, and applications using

MQOO_BIND_AS_Q_DEF where the queue has been defined with DEFBIND(NOTFIXED), fail on the next attempted MQPUT or MQPUT1 call. The reason code MQRC_UNKNOWN_OBJECT_NAME is returned.

You do not have to perform Step 1 first, but if you do not, perform it instead after Step 4.

2. Stop all applications that have access to cluster queue.

   Stop all applications that have access to cluster queues. If you do not, some cluster information might remain on the local queue manager when you refresh the cluster in Step 5. This information is removed when all applications have stopped and the cluster channels have disconnected.

3. Remove the repository attribute from the full repository queue managers.

   On both CHICAGO and CHICAGO2, modify the queue manager definitions to remove the repository attribute. To do this issue the command:

```
ALTER QMGR REPOS(' ')
```

   The queue managers inform the other queue managers in the cluster that they no longer hold the full repositories. When the other queue managers receive this information, you see a message indicating that the full repository has ended. You also see one or more messages indicating that there are no longer any repositories available for the cluster CHNSTORE.

4. Remove cluster channels.

   On CHICAGO remove the cluster channels:

```
ALTER CHANNEL(CHNSTORE.CHICAGO2) CHLTYPE(CLUSSDR) CLUSTER(' ')
ALTER CHANNEL(CHNSTORE.CHICAGO) CHLTYPE(CLUSRCVR) CLUSTER(' ')
```

   Note: It is important to issue the CLUSSDR command first, then CLUSRCVR command. Do not issue the CLUSRCVR command first, then the CLUSSDR command. Doing so, creates indoubt channels that have a STOPPED status. You then need to issue a START CHANNEL command to recover the stopped channels; for example, START CHANNEL(CHNSTORE.CHICAGO).

   You see messages indicating that there are no repositories for the cluster CHNSTORE.

   If you did not remove the cluster queues as described in Step 1, do so now.

5. Stop cluster channels.

   On CHICAGO stop the cluster channels with the following commands:

```
STOP CHANNEL(CHNSTORE.CHICAGO2)
STOP CHANNEL(CHNSTORE.CHICAGO)
```

6. Repeat steps 4 and 5 for each queue manager in the cluster.
7. Stop the cluster channels, then remove all definitions for the cluster channels and cluster queues from each queue manager.
8. Optional: Clear the cached cluster information held by the queue manager.

   Although the queue managers are no longer members of the cluster, they each retain a cached copy of information about the cluster. If you want to remove this data, see task “Restoring a queue manager to its pre-cluster state” on page 289.

9. Replace the remote-queue definitions for the INVENTQ

   So that the network can continue to function, replace the remote queue definition for the INVENTQ at every queue manager.

10. Tidy up the cluster.

    Delete any queue or channel definitions no longer required.

Related tasks

Adding a new, interconnected cluster

Add a new cluster that shares some queue managers with an existing cluster.

Creating two-overlapping clusters with a gateway queue manager

Follow the instructions in the task to construct overlapping clusters with a gateway queue manager. Use the clusters as a starting point for the following examples of isolating messages to one application from messages to other applications in a cluster.

About this task

The example cluster configuration used to illustrate isolating cluster message traffic is shown in Figure 48 on page 263. The example is described in Clustering: Application isolation using multiple cluster transmission queues.

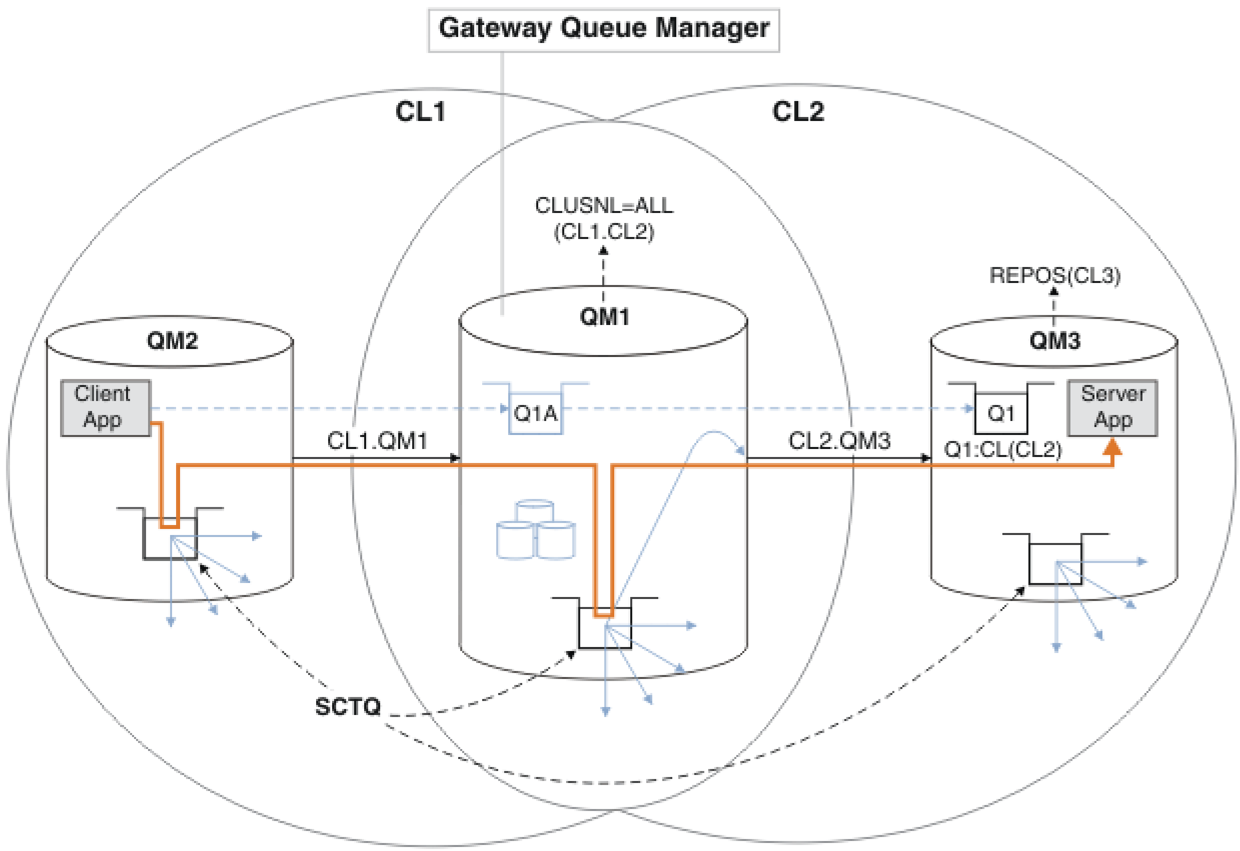

Figure 48. Client-server application deployed to hub and spoke architecture using IBM MQ clusters

To make the number of steps to construct the example as few as possible, the configuration is kept simple, rather than realistic. The example might represent the integration of two clusters created by two separate organizations. For a more realistic scenario, see Clustering: Planning how to configure cluster transmission queues.

Follow the steps to construct the clusters. The clusters are used in the following examples of isolating the message traffic from the client application to the server application.

The instructions add a couple of extra queue managers so that each cluster has two repositories. The gateway queue manager is not used as a repository for performance reasons.

Procedure

1. Create and start the queue managers `QM1`, `QM2`, `QM3`, `QM4`, `QM5`.

```
crtmqm -sax -u SYSTEM.DEAD.LETTER.QUEUE QM n
strmqm QmgrName
```

Note: QM4 and QM5 are the backup full repositories for the clusters.

2. Define and start listeners for each of the queue managers.

```
*... On QM n
DEFINE LISTENER(TCP141 n) TRPTYPE(TCP) IPADDR(hostname) PORT(141 n) CONTROL(QMGR) REPLACE
START LISTENER(TCP141 n)
```

3. Create a cluster name list for all of the clusters.

```
*... On QM1
DEFINE NAMELIST(ALL) NAMES(CL1, CL2) REPLACE
```

4. Make QM2 and QM4 full repositories for CL1, QM3 and QM5 full repositories for CL2.
   a) For CL1:

```
*... On QM2 and QM4
ALTER QMGR REPOS(CL1) DEFCLXQ(SCTQ)
```

   b) For CL2:

```
*... On QM3 and QM5
ALTER QMGR REPOS(CL2) DEFCLXQ(SCTQ)
```

5. Add the cluster-sender and cluster-receiver channels for each queue manager and cluster.

   Run the following commands on QM2, QM3, QM4 and QM5, where *c*, *n*, and *m* take the values shown in Table 25 on page 264 for each queue manager:

Table 25. Parameter values for creating clusters 1 and 2

Queue manager	**Cluster** ***c***	**Other repository** ***n***	**This repository** ***m***
QM2	1	4	2
QM4	1	2	4
QM3	2	5	3
QM5	2	3	5

```
*... On QM m
DEFINE CHANNEL(CL c.QM n) CHLTYPE(CLUSSDR) CONNAME('localhost(141 n)') CLUSTER(CL c) REPLACE
DEFINE CHANNEL(CL c.QM m) CHLTYPE(CLUSRCVR) CONNAME('localhost(141 m)') CLUSTER(CL c) REPLACE
```

6. Add the gateway queue manager, QM1, to each of the clusters.

```
*... On QM1
DEFINE CHANNEL(CL1.QM2) CHLTYPE(CLUSSDR) CONNAME('localhost(1412)') CLUSTER(CL1) REPLACE
DEFINE CHANNEL(CL1.QM1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1411)') CLUSTER(CL1) REPLACE
DEFINE CHANNEL(CL2.QM3) CHLTYPE(CLUSSDR) CONNAME('localhost(1413)') CLUSTER(CL2) REPLACE
DEFINE CHANNEL(CL2.QM1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1411)') CLUSTER(CL2) REPLACE
```

7. Add the local queue Q1 to queue manager QM3 in cluster CL2.

```
*... On QM3
DEFINE QLOCAL(Q1) CLUSTER(CL2) REPLACE
```

8. Add the clustered queue manager alias Q1A to the gateway queue manager.

```
*... On QM1
DEFINE QALIAS(Q1A) CLUSNL(ALL) TARGET(Q1) TARGTYPE(QUEUE) DEFBIND(NOTFIXED) REPLACE
```

Note: Applications using the queue manager alias on any other queue manager but QM1, must specify DEFBIND(NOTFIXED) when they open the alias queue. **DEFBIND** specifies whether the routing information in the message header is fixed when the queue is opened by the application. If it is set to the default value, OPEN, messages are routed to Q1@QM1. Q1@QM1 does not exist, so messages from other queue managers end up on a dead letter queue. By setting the queue attribute to DEFBIND(NOTFIXED), applications such as **amqsput**, which default to the queue setting of **DEFBIND**, behave in the correct way.

9. Add the cluster queue manager alias definitions for all the clustered queue managers to the gateway queue manager, QM1.

```
*... On QM1
DEFINE QREMOTE(QM2) RNAME(' ') RQMNAME(QM2) CLUSNL(ALL) REPLACE
DEFINE QREMOTE(QM3) RNAME(' ') RQMNAME(QM3) CLUSNL(ALL) REPLACE
```

Tip: The queue manager alias definitions on the gateway queue manager transfer messages that refer to a queue manager in another cluster; see Clustered queue manager aliases.

What to do next

1. Test the queue alias definition by sending a message from QM2 to Q1 on QM3 using the queue alias definition Q1A.
   a. Run the sample program **amqsput** on QM2 to put a message.

   ```
   C:\IBM\MQ>amqsput Q1A QM2
   Sample AMQSPUT0 start
   target queue is Q1A
   Sample request message from QM2 to Q1 using Q1A

   Sample AMQSPUT0 end
   ```

   b. Run the sample program **amqsget** to get the message from Q1 on QM3

   ```
   C:\IBM\MQ>amqsget Q1 QM3
   Sample AMQSGET0 start
   message <Sample request message from QM2 to Q1 using Q1A>
   no more messages
   Sample AMQSGET0 end
   ```

2. Test the queue manager alias definitions by sending a request message and receiving a reply message on a temporary-dynamic reply queue.

   The diagram shows the path taken by the reply message back to a temporary dynamic queue, which is called RQ. The server application, connected to QM3, opens the reply queue using the queue manager name QM2. The queue manager name QM2 is defined as a clustered queue manager alias on QM1. QM3 routes the reply message to QM1. QM1 routes the message to QM2.

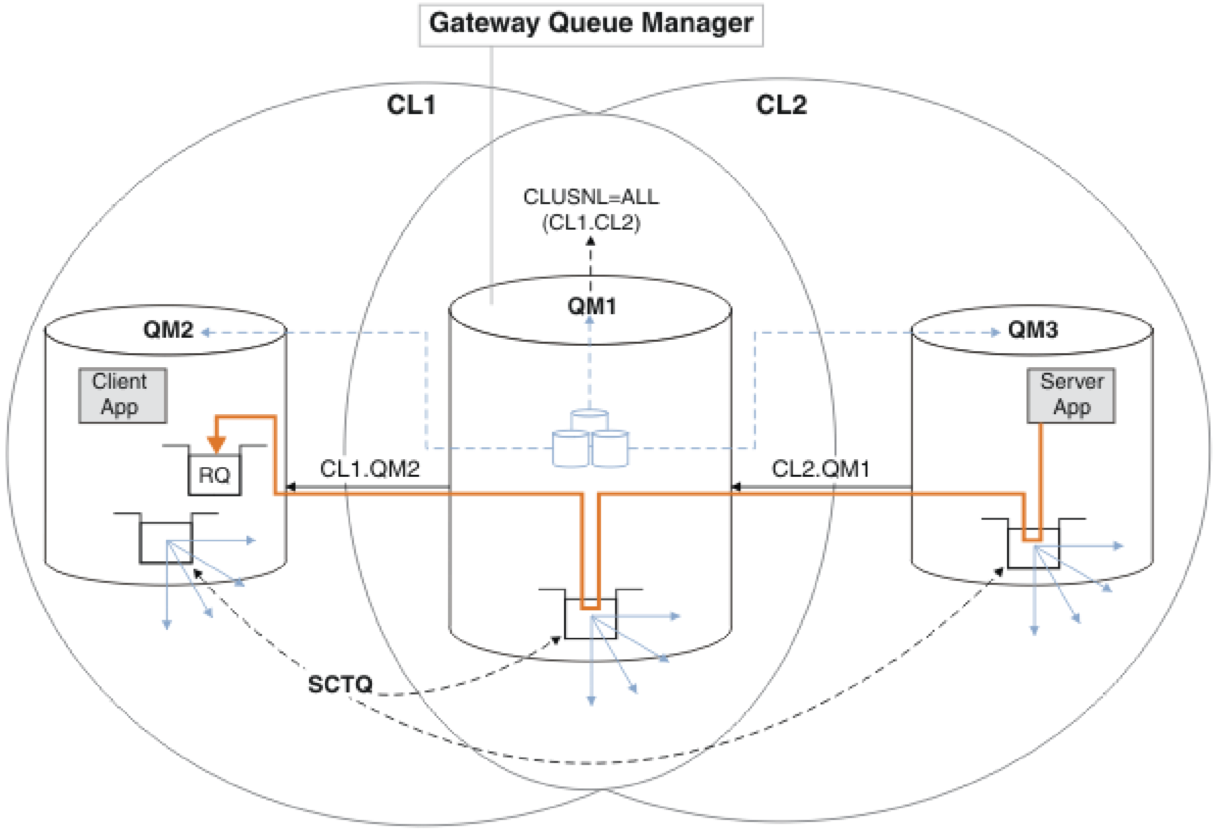

Figure 49. Using a queue manager alias to return the reply message to a different cluster

The way the routing works is as follows. Every queue manager in each cluster has a queue manager alias definition on QM1. The aliases are clustered in all the clusters. The grey dashed arrows from each of the aliases to a queue manager show that each queue manager alias is resolved to a real queue manager in at least one of the clusters. In this case, the QM2 alias is clustered in both cluster CL1 and CL2, and is resolved to the real queue manager QM2 in CL1. The server application creates the reply message using the reply to queue name RQ, and reply to queue manager name QM2. The message is routed to QM1 because the queue manager alias definition QM2 is defined on QM1 in cluster CL2 and queue manager QM2 is not in cluster CL2. As the message cannot be sent to the target queue manager, it is sent to the queue manager that has the alias definition.

QM1 places the message on the cluster transmission queue on QM1 for transferal to QM2. QM1 routes the message to QM2 because the queue manager alias definition on QM1 for QM2 defines QM2 as the real target queue manager. The definition is not circular, because alias definitions can refer only to real definitions; the alias cannot point to itself. The real definition is resolved by QM1, because both QM1 and QM2 are in the same cluster, CL1. QM1 finds out the connection information for QM2 from the repository for CL1, and routes the message to QM2. For the message to be rerouted by QM1, the server application must have opened the reply queue with the option DEFBIND set to MQBND_BIND_NOT_FIXED. If the server application had opened the reply queue with the option MQBND_BIND_ON_OPEN, the message is not rerouted and ends up on a dead letter queue.

a. Create a clustered request queue with a trigger on QM3.

```
*... On QM3
DEFINE QLOCAL(QR) CLUSTER(CL2) TRIGGER INITQ(SYSTEM.DEFAULT.INITIATION.QUEUE)
PROCESS(ECHO) REPLACE
```

b. Create a clustered queue alias definition of QR on the gateway queue manager, QM1.

```
*... On QM1
DEFINE QALIAS(QRA) CLUSNL(ALL) TARGET(QR) TARGTYPE(QUEUE) DEFBIND(NOTFIXED) REPLACE
```

c. Create a process definition to start the sample echo program **amqsech** on QM3.

```
*... On QM3
DEFINE PROCESS(ECHO) APPLICID(AMQSECH) REPLACE
```

d. Create a model queue on QM2 for the sample program **amqsreq** to create the temporary-dynamic reply queue.

```
*... On QM2
DEFINE QMODEL(SYSTEM.SAMPLE.REPLY) REPLACE
```

e. Test the queue manager alias definition by sending a request from QM2 to QR on QM3 using the queue alias definition QRA.

i) Run the trigger monitor program on QM3.

```
runmqtrm -m QM3
```

The output is

```
C:\IBM\MQ>runmqtrm -m QM3
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
01/02/2012  16:17:15: IBM MQ trigger monitor started.

__________________________________________________
01/02/2012  16:17:15: Waiting for a trigger message
```

ii) Run the sample program **amqsreq** on QM2 to put a request and wait for a reply.

```
C:\IBM\MQ>amqsreq QRA QM2
Sample AMQSREQ0 start
server queue is QRA
replies to 4F2961C802290020
A request message from QM2 to QR on QM3

response <A request message from QM2 to QR on QM3>
no more replies
Sample AMQSREQ0 end
```

Related concepts

Access control and multiple cluster transmission queues

Clustering: Application isolation using multiple cluster transmission queues

Related tasks

Clustering: Planning how to configure cluster transmission queues

"Adding a queue manager to a cluster: separate transmission queues" on page 240
Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using multiple cluster transmission queues.

Adding a remote queue definition to isolate messages sent from a gateway queue manager

Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses a clustered queue remote definition, and a separate sender channel and transmission queue.

Before you begin

Construct the overlapping clusters shown in Client-server application deployed to hub and spoke architecture using IBM MQ clusters in "Creating two-overlapping clusters with a gateway queue manager" on page 263 by following the steps in that task.

About this task

The solution uses distributed queuing to separate the messages for the `Server App` application from other message traffic on the gateway queue manager. You must define a clustered remote queue definition on QM1 to divert the messages to a different transmission queue, and a different channel. The remote queue definition must include a reference to the specific transmission queue that stores messages only for Q1 on QM3. In Figure 50 on page 268, the cluster queue alias Q1A is supplemented by a remote queue definition Q1R, and a transmission queue and sender-channel added.

In this solution, any reply messages are returned using the common `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

The advantage of this solution is that it is easy to separate traffic for multiple destination queues on the same queue manager, in the same cluster. The disadvantage of the solution is that you cannot use cluster workload balancing between multiple copies of Q1 on different queue managers. To overcome this disadvantage, see "Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager" on page 270. You also have to manage the switch from one transmission queue to the other.

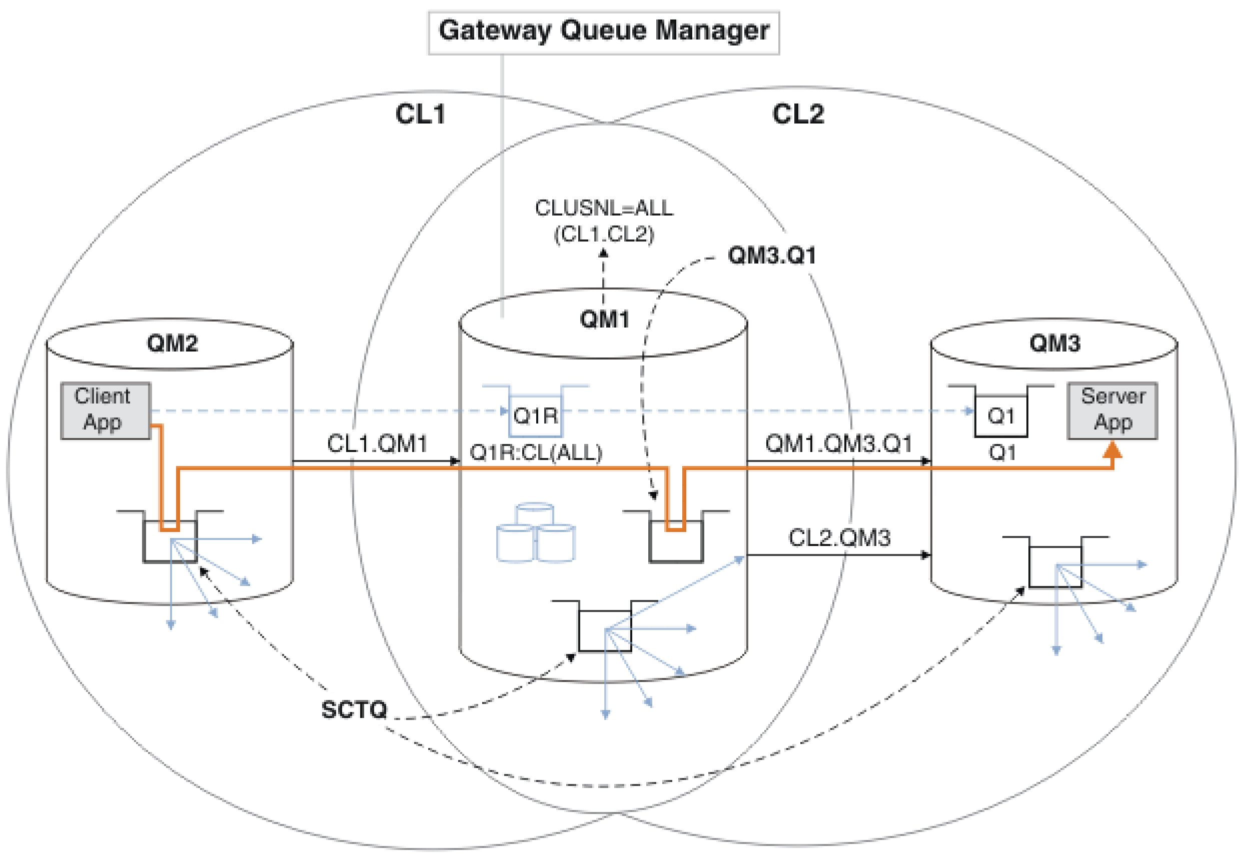

Figure 50. Client-server application deployed to hub and spoke cluster architecture using remote queue definitions

Procedure

1. Create a channel to separate the message traffic for Q1 from the gateway queue manager

a) Create a sender channel on the gateway queue manager, QM1, to the target queue manager, QM3.

```
DEFINE CHANNEL(QM1.QM3.Q1) CHLTYPE(SDR) CONNAME(QM3HostName(1413)) XMITQ(QM3.Q1) REPLACE
```

b) Create a receiver channel on the target queue manager, QM3.

```
DEFINE CHANNEL(QM1.QM3.Q1) CHLTYPE(RCVR) REPLACE
```

2. Create a transmission queue on the gateway queue manager for message traffic to Q1

```
DEFINE QLOCAL(QM3.Q1) USAGE(XMITQ) REPLACE
START CHANNEL(QM1.QM3.Q1)
```

Starting the channel that is associated with the transmission queue, associates the transmission queue with the channel. The channel starts automatically, once the transmission queue has been associated with the channel.

3. Supplement the clustered queue alias definition for Q1 on the gateway queue manager with a clustered remote queue definition.

```
DEFINE QREMOTE CLUSNL(ALL) RNAME(Q1) RQMNAME(QM3) XMITQ(QM3.Q1) REPLACE
```

What to do next

Test the configuration by sending a message to Q1 on QM3 from QM2 using the clustered queue remote definition Q1R on the gateway queue manager QM1.

1. Run the sample program **amqsput** on QM2 to put a message.

```
C:\IBM\MQ>amqsput Q1R QM2
Sample AMQSPUT0 start
target queue is Q1R
Sample request message from QM2 to Q1 using Q1R

Sample AMQSPUT0 end
```

2. Run the sample program **amqsget** to get the message from Q1 on QM3

```
C:\IBM\MQ>amqsget Q1 QM3
Sample AMQSGET0 start
message <Sample request message from QM2 to Q1 using Q1R>
no more messages
Sample AMQSGET0 end
```

Related concepts

Clustering: Application isolation using multiple cluster transmission queues

Access control and multiple cluster transmission queues

Related tasks

Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses an additional cluster transmission queue to separate message traffic to a single queue manager in a cluster.

Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same

transmission queue or channels as other cluster messages. The solution uses an additional cluster to isolate the messages to a particular cluster queue.

Changing the default to separate cluster transmission queues to isolate message traffic
You can change the default way a queue manager stores messages for a clustered queue or topic on a transmission queue. Changing the default provides you with a way to isolate cluster messages on a gateway queue manager.

Clustering: Planning how to configure cluster transmission queues

"Adding a queue manager to a cluster: separate transmission queues" on page 240
Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using multiple cluster transmission queues.

Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager

Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses an additional cluster transmission queue to separate message traffic to a single queue manager in a cluster.

Before you begin

1. The gateway queue manager must be on Version 7.5, or later.
2. Construct the overlapping clusters shown in Client-server application deployed to hub and spoke architecture using IBM MQ clusters in "Creating two-overlapping clusters with a gateway queue manager" on page 263 by following the steps in that task.

About this task

On the gateway queue manager, QM1, add a transmission queue and set its queue attribute CLCHNAME. Set CLCHNAME to the name of the cluster-receiver channel on QM3 ; see Figure 51 on page 271.

This solution has a number of advantages over the solution described in "Adding a remote queue definition to isolate messages sent from a gateway queue manager" on page 267:

- It requires fewer additional definitions.
- It supports workload balancing between multiple copies of the target queue, Q1, on different queue managers in the same cluster, CL2.
- The gateway queue manager switches automatically to the new configuration when the channel restarts without loosing any messages.
- The gateway queue manager continues to forward messages in the same order as it received them. It does so, even if the switch takes place with messages for the queue Q1 at QM3 still on SYSTEM.CLUSTER.TRANSMIT.QUEUE.

The configuration to isolate cluster message traffic in Figure 51 on page 271 does not result in as great an isolation of traffic as the configuration using remote queues in "Adding a remote queue definition to isolate messages sent from a gateway queue manager" on page 267. If the queue manager QM3 in CL2 is hosting a number of different cluster queues and server applications, all those queues share the cluster channel, CL2.QM3, connecting QM1 to QM3. The additional flows are illustrated in Figure 51 on page 271 by the gray arrow representing potential cluster message traffic from the SYSTEM.CLUSTER.TRANSMIT.QUEUE to the cluster-sender channel CL2.QM3.

The remedy is to restrict the queue manager to hosting one cluster queue in a particular cluster. If the queue manager is already hosting a number of cluster queues, then to meet this restriction, you must either create another queue manager, or create another cluster; see "Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager" on page 273.

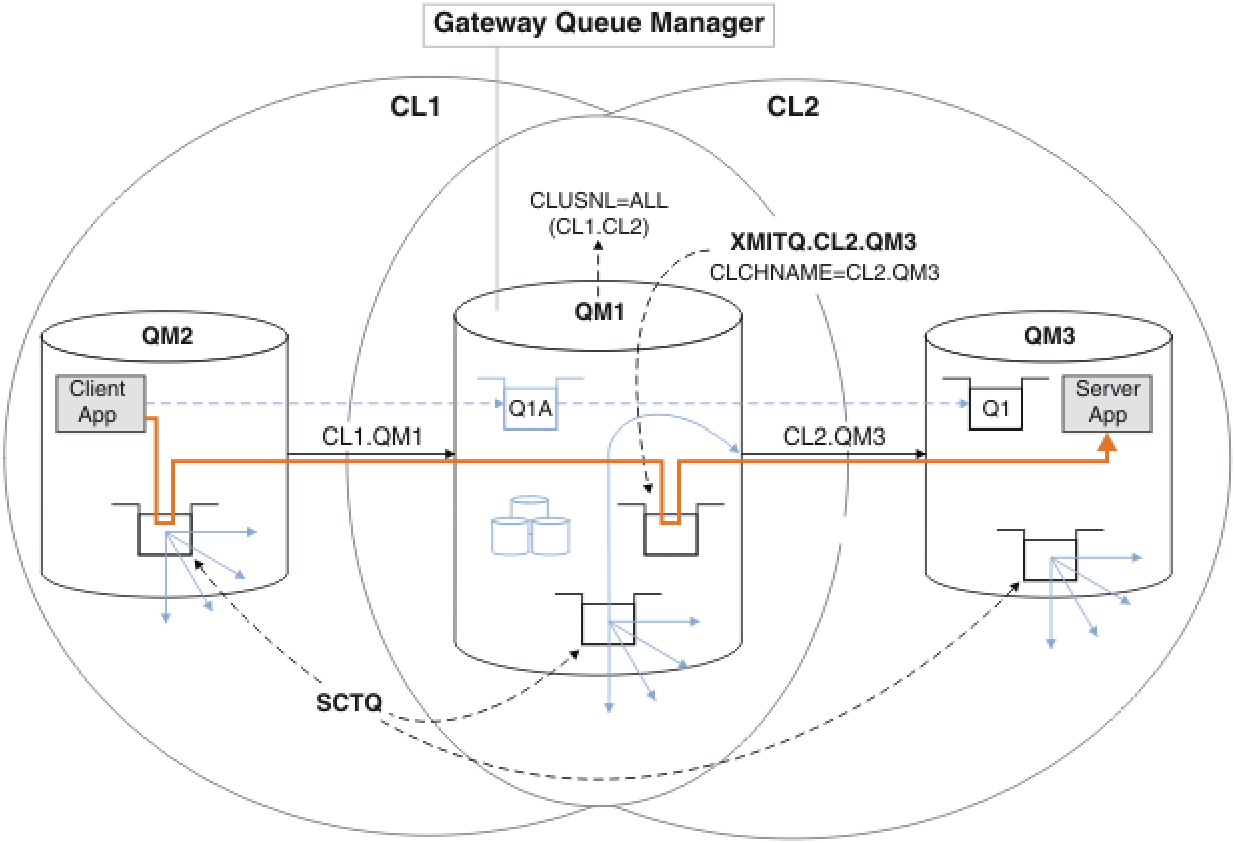

Figure 51. Client-server application deployed to hub and spoke architecture using an additional cluster transmission queue.

Procedure

1. Create an additional cluster transmission queue for the cluster-sender channel CL2.QM3 on the gateway queue manager, QM1.

```
*... on QM1
DEFINE QLOCAL(XMITQ.CL2.QM3) USAGE(XMITQ) CLCHNAME(CL2.QM3)
```

2. Switch to using the transmission queue, XMITQ.CL2.QM3.

   a) Stop the cluster-sender channel CL2.QM3.

   ```
   *... On QM1
   STOP CHANNEL(CL2.QM3)
   ```

   The response is that the command is accepted:

   ```
   AMQ8019: Stop IBM MQ channel accepted.
   ```

   b) Check that the channel CL2.QM3 is stopped

   If the channel does not stop, you can run the **STOP CHANNEL** command again with the FORCE option. An example of setting the FORCE option would be if the channel does not stop, and you cannot restart the other queue manager to synchronize the channel.

   ```
   *... On QM1
   start
   ```

The response is a summary of the channel status

```
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)                 CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))         CURRENT
RQMNAME(QM3)                     STATUS(STOPPED)
SUBSTATE(MQGET)                  XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
```

c) Start the channel, CL2.QM3.

```
*... On QM1
START CHANNEL(CL2.QM3)
```

The response is that the command is accepted:

```
AMQ8018: Start IBM MQ channel accepted.
```

d) Check the channel started.

```
*... On QM1
DISPLAY CHSTATUS(CL2.QM3)
```

The response is a summary of the channel status:

```
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)                       CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))               CURRENT
RQMNAME(QM3)                           STATUS(RUNNING)
SUBSTATE(MQGET)                        XMITQ(XMITQ.CL2.QM3)
```

e) Check the transmission queue was switched.

Monitor the gateway queue manager error log for the message " AMQ7341 The `transmission queue for channel CL2.QM3 is XMITQ.CL2.QM3` ".

What to do next

Test the separate transmission queue by sending a message from QM2 to Q1 on QM3 using the queue alias definition Q1A

1. Run the sample program **amqsput** on QM2 to put a message.

```
C:\IBM\MQ>amqsput Q1A QM2
Sample AMQSPUT0 start
target queue is Q1A
Sample request message from QM2 to Q1 using Q1A

Sample AMQSPUT0 end
```

2. Run the sample program **amqsget** to get the message from Q1 on QM3

```
C:\IBM\MQ>amqsget Q1 QM3
Sample AMQSGET0 start
message <Sample request message from QM2 to Q1 using Q1A>
no more messages
Sample AMQSGET0 end
```

Related concepts

Access control and multiple cluster transmission queues

Clustering: Application isolation using multiple cluster transmission queues

"Working with cluster transmission queues and cluster-sender channels" on page 221
Messages between clustered queue managers are stored on cluster transmission queues and forwarded by cluster-sender channels. At any point in time, a cluster-sender channel is associated with one transmission queue. If you change the configuration of the channel, it might switch to a different transmission queue next time it starts. The processing of this switch is automated, and transactional.

Related tasks

Adding a remote queue definition to isolate messages sent from a gateway queue manager
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses a clustered queue remote definition, and a separate sender channel and transmission queue.

Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses an additional cluster to isolate the messages to a particular cluster queue.

Changing the default to separate cluster transmission queues to isolate message traffic
You can change the default way a queue manager stores messages for a clustered queue or topic on a transmission queue. Changing the default provides you with a way to isolate cluster messages on a gateway queue manager.

Clustering: Planning how to configure cluster transmission queues

"Adding a queue manager to a cluster: separate transmission queues" on page 240
Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using multiple cluster transmission queues.

Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager

Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses an additional cluster to isolate the messages to a particular cluster queue.

Before you begin

The steps in the task are written to modify the configuration illustrated in Figure 51 on page 271.

1. The gateway queue manager must be on Version 7.5, or later.
2. Construct the overlapping clusters shown in Client-server application deployed to hub and spoke architecture using IBM MQ clusters in "Creating two-overlapping clusters with a gateway queue manager" on page 263 by following the steps in that task.
3. Do the steps in Figure 51 on page 271 in "Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager" on page 270 to create the solution without the additional cluster. Use this as a base for the steps in this task.

About this task

The solution to isolating message traffic to a single application in "Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager" on page 270 works if the target cluster queue is the only cluster queue on a queue manager. If it is not, you have two choices. Either move the queue to a different queue manager, or create a cluster that isolates the queue from other cluster queues on the queue manager.

This task takes you through the steps to add a cluster to isolate the target queue. The cluster is added just for that purpose. In practice, approach the task of isolating certain applications systematically when you are in the process of designing clusters and cluster naming schemes. Adding a cluster each time a queue requires isolation might end up with many clusters to manage. In this task, you change the configuration in "Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager" on page 270 by adding a cluster CL3 to isolate Q1 on QM3. Applications continue to run throughout the change.

The new and changed definitions are highlighted in Figure 52 on page 274. The summary of the changes is as follows: Create a cluster, which means you must also create a new full cluster repository. In the example, QM3 is made one of the full repositories for CL3. Create cluster-sender and cluster-receiver channels for QM1 to add the gateway queue manager to the new cluster. Change the definition of Q1 to switch it to CL3. Modify the cluster namelist on the gateway queue manager, and add a cluster transmission queue to use the new cluster channel. Finally, switch the queue alias Q1A to the new cluster namelist.

IBM MQ cannot transfer messages from the transmission queue XMITQ.CL2.QM3 that you added in "Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager" on page 270 to the new transmission queue XMITQ.CL3.QM3, automatically. It can transfer messages automatically only if both transmission queues are served by the same cluster-sender channel. Instead, the task describes one way to perform the switch manually, which might be appropriate to you. When the transfer is completed, you have the option of reverting to using the default cluster transmission queue for other CL2 cluster queues on QM3. Or you can continue to use XMITQ.CL2.QM3. If you decide to revert to a default cluster transmission queue, the gateway queue manager manages the switch for you automatically.

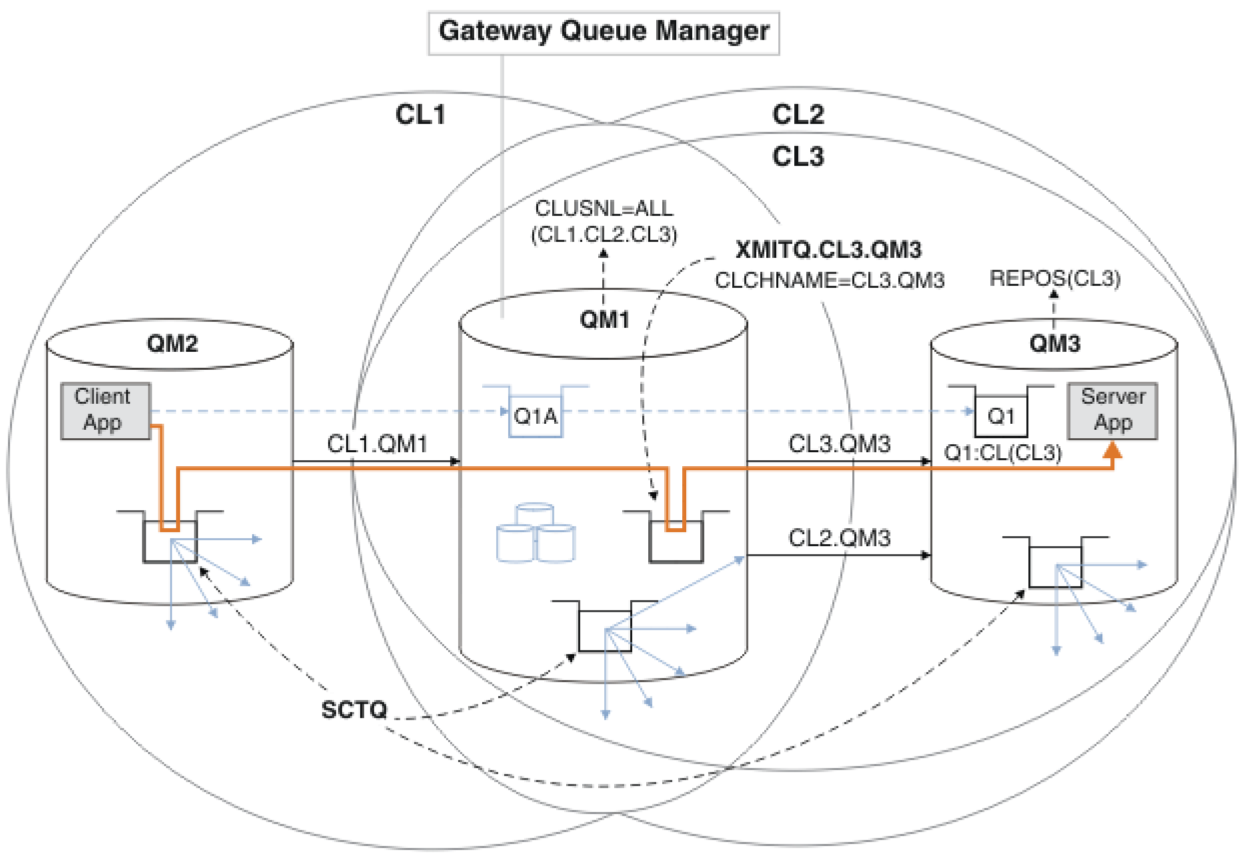

Figure 52. Using an additional cluster to separate message traffic in the gateway queue manager that goes to one of a number of cluster queues on the same queue manager

Procedure

1. Alter the queue managers QM3 and QM5 to make them repositories for both CL2 and CL3.

   To make a queue manager a member of multiple clusters, it must use a cluster name list to identify the clusters it is a member of.

   ```
   *... On QM3 and QM5
   DEFINE NAMELIST(CL23) NAMES(CL2, CL3) REPLACE
   ALTER QMGR REPOS(' ') REPOSNL(CL23)
   ```

2. Define the channels between the queue managers QM3 and QM5 for CL3.

   ```
   *... On QM3
   DEFINE CHANNEL(CL3.QM5) CHLTYPE(CLUSSDR) CONNAME('localhost(1415)') CLUSTER(CL3) REPLACE
   DEFINE CHANNEL(CL3.QM3) CHLTYPE(CLUSRCVR) CONNAME('localhost(1413)') CLUSTER(CL3) REPLACE

   *... On QM5
   DEFINE CHANNEL(CL3.QM3) CHLTYPE(CLUSSDR) CONNAME('localhost(1413)') CLUSTER(CL3) REPLACE
   DEFINE CHANNEL(CL3.QM5) CHLTYPE(CLUSRCVR) CONNAME('localhost(1415)') CLUSTER(CL3) REPLACE
   ```

3. Add the gateway queue manager to CL3.

   Add the gateway queue manager by adding QM1 to CL3 as a partial repository. Create a partial repository by adding cluster-sender and cluster-receiver channels to QM1.

   Also, add CL3 to the name list of all clusters connected to the gateway queue manager.

   ```
   *... On QM1
   DEFINE CHANNEL(CL3.QM3) CHLTYPE(CLUSSDR) CONNAME('localhost(1413)') CLUSTER(CL3) REPLACE
   DEFINE CHANNEL(CL3.QM1) CHLTYPE(CLUSRCVR) CONNAME('localhost(1411)') CLUSTER(CL3) REPLACE
   ALTER NAMELIST(ALL) NAMES(CL1, CL2, CL3)
   ```

4. Add a cluster transmission queue to the gateway queue manager, QM1, for messages going to CL3 on QM3.

   Initially, stop the cluster-sender channel transferring messages from the transmission queue until you are ready to switch transmission queues.

   ```
   *... On QM1
   DEFINE QLOCAL(XMITQ.CL3.QM3) USAGE(XMITQ) CLCHNAME(CL3.QM3) GET(DISABLED) REPLACE
   ```

5. Drain messages from the existing cluster transmission queue XMITQ.CL2.QM3.

   This subprocedure is intended to preserve the order of messages in Q1 to match the order they arrived at the gateway queue manager. With clusters, message ordering is not fully guaranteed, but is likely. If guaranteed message ordering is required, applications must define the order of messages; see The order in which messages are retrieved from a queue.

   a) Change the target queue Q1 on QM3 from CL2 to CL3.

      ```
      *... On QM3
      ALTER QLOCAL(Q1) CLUSTER(CL3)
      ```

   b) Monitor XMITQ.CL3.QM3 until messages start to be delivered to it.

      Messages start to be delivered to XMITQ.CL3.QM3 when the switch of Q1 to CL3 is propagated to the gateway queue manager.

      ```
      *... On QM1
      DISPLAY QUEUE(XMITQ.CL3.QM3) CURDEPTH
      ```

   c) Monitor XMITQ.CL2.QM3 until it has no messages waiting to be delivered to Q1 on QM3.

      Note: XMITQ.CL2.QM3 might be storing messages for other queues on QM3 that are members of CL2, in which case the depth might not go to zero.

```
*... On QM1
DISPLAY QUEUE(XMITQ.CL2.QM3) CURDEPTH
```

d) Enable get from the new cluster transmission queue, XMITQ.CL3.QM3

```
*... On QM1
ALTER QLOCAL(XMITQ.CL3.QM3) GET(ENABLED)
```

6. Remove the old cluster transmission queue, XMITQ.CL2.QM3, if it is no longer required.

   Messages for cluster queues in CL2 on QM3 revert to using the default cluster transmission queue on the gateway queue manager, QM1. The default cluster transmission queue is either SYSTEM.CLUSTER.TRANSMIT.QUEUE, or SYSTEM.CLUSTER.TRANSMIT.CL2.QM3. Which one depends on whether the value of the queue manager attribute **DEFCLXQ** on QM1 is SCTQ or CHANNEL. The queue manager transfers messages from XMITQ.CL2.QM3 automatically when the cluster-sender channel CL2.QM3 next starts.

   a) Change the transmission queue, XMITQ.CL2.QM3, from being a cluster transmission queue to being a normal transmission queue.

      This breaks the association of the transmission queue with any cluster-sender channels. In response, IBM MQ automatically transfers messages from XMITQ.CL2.QM3 to the default cluster transmission queue when the cluster-sender channel is next started. Until then, messages for CL2 on QM3 continue to be placed on XMITQ.CL2.QM3.

```
*... On QM1
ALTER QLOCAL(XMITQ.CL2.QM3) CLCHNAME(' ')
```

   b) Stop the cluster-sender channel CL2.QM3.

      Stopping and restarting the cluster-sender channel initiates the transfer of messages from XMITQ.CL2.QM3 to the default cluster transmission queue. Typically you would stop and start the channel manually to start the transfer. The transfer starts automatically if the channel restarts after shutting down on the expiry of its disconnect interval.

```
*... On QM1
STOP CHANNEL(CL2.QM3)
```

      The response is that the command is accepted:

```
AMQ8019: Stop IBM MQ channel accepted.
```

   c) Check that the channel CL2.QM3 is stopped

      If the channel does not stop, you can run the **STOP CHANNEL** command again with the FORCE option. An example of setting the FORCE option would be if the channel does not stop, and you cannot restart the other queue manager to synchronize the channel.

```
*... On QM1
DISPLAY CHSTATUS(CL2.QM3)
```

      The response is a summary of the channel status

```
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)                    CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))            CURRENT
RQMNAME(QM3)                        STATUS(STOPPED)
SUBSTATE(MQGET)                     XMITQ(XMITQ.CL2.QM3)
```

   d) Start the channel, CL2.QM3.

```
*... On QM1
START CHANNEL(CL2.QM3)
```

The response is that the command is accepted:

```
AMQ8018: Start IBM MQ channel accepted.
```

e) Check the channel started.

```
*... On QM1
DISPLAY CHSTATUS(CL2.QM3)
```

The response is a summary of the channel status:

```
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)      CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))  CURRENT
RQMNAME(QM3)        STATUS(RUNNING)
SUBSTATE(MQGET)      XMITQ(SYSTEM.CLUSTER.TRANSMIT. QUEUE|CL2.QM3)
```

f) Monitor the gateway queue manager error log for the message " AMQ7341 The `transmission queue for channel CL2.QM3 is SYSTEM.CLUSTER.TRANSMIT.` *`QUEUE|CL2.QM3`* ".

g) Delete the cluster transmission queue, `XMITQ.CL2.QM3`.

```
*... On QM1
DELETE QLOCAL(XMITQ.CL2.QM3)
```

What to do next

Test the separately clustered queue by sending a message from QM2 to Q1 on QM3 using the queue alias definition Q1A

1. Run the sample program **`amqsput`** on QM2 to put a message.

```
C:\IBM\MQ>amqsput Q1A QM2
Sample AMQSPUT0 start
target queue is Q1A
Sample request message from QM2 to Q1 using Q1A

Sample AMQSPUT0 end
```

2. Run the sample program **`amqsget`** to get the message from Q1 on QM3

```
C:\IBM\MQ>amqsget Q1 QM3
Sample AMQSGET0 start
message <Sample request message from QM2 to Q1 using Q1A>
no more messages
Sample AMQSGET0 end
```

Related concepts

Access control and multiple cluster transmission queues

Clustering: Application isolation using multiple cluster transmission queues

“Working with cluster transmission queues and cluster-sender channels” on page 221

Messages between clustered queue managers are stored on cluster transmission queues and forwarded by cluster-sender channels. At any point in time, a cluster-sender channel is associated with one transmission queue. If you change the configuration of the channel, it might switch to a different transmission queue next time it starts. The processing of this switch is automated, and transactional.

Related tasks

Adding a remote queue definition to isolate messages sent from a gateway queue manager
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses a clustered queue remote definition, and a separate sender channel and transmission queue.

Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses an additional cluster transmission queue to separate message traffic to a single queue manager in a cluster.

Changing the default to separate cluster transmission queues to isolate message traffic
You can change the default way a queue manager stores messages for a clustered queue or topic on a transmission queue. Changing the default provides you with a way to isolate cluster messages on a gateway queue manager.

Clustering: Planning how to configure cluster transmission queues

"Adding a queue manager to a cluster: separate transmission queues" on page 240
Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using multiple cluster transmission queues.

Changing the default to separate cluster transmission queues to isolate message traffic

You can change the default way a queue manager stores messages for a clustered queue or topic on a transmission queue. Changing the default provides you with a way to isolate cluster messages on a gateway queue manager.

Before you begin

1. The gateway queue manager must be on Version 7.5, or later.
2. Construct the overlapping clusters shown in Client-server application deployed to hub and spoke architecture using IBM MQ clusters in "Creating two-overlapping clusters with a gateway queue manager" on page 263 by following the steps in that task.

About this task

To implement the architecture with multiple clusters queue, your gateway queue manager must be on Version 7.5, or later. All you do to use multiple cluster transmission queues is to change the default cluster transmission queue type on the gateway queue manager. Change the value of the queue manager attribute **DEFCLXQ** on `QM1` from `SCTQ` to `CHANNEL` ; see Figure 53 on page 279. The diagram shows one message flow. For flows to other queue managers, or to other clusters, the queue manager creates additional permanent dynamic cluster transmission queues. Each cluster-sender channel transfers messages from a different cluster transmission queue.

The change does not take effect immediately, unless you are connecting the gateway queue manager to clusters for the first time. The task includes steps for the typical case of managing a change to an existing configuration. To set up a queue manager to use separate cluster transmission queues when it first joins a cluster; see "Adding a queue manager to a cluster: separate transmission queues" on page 240.

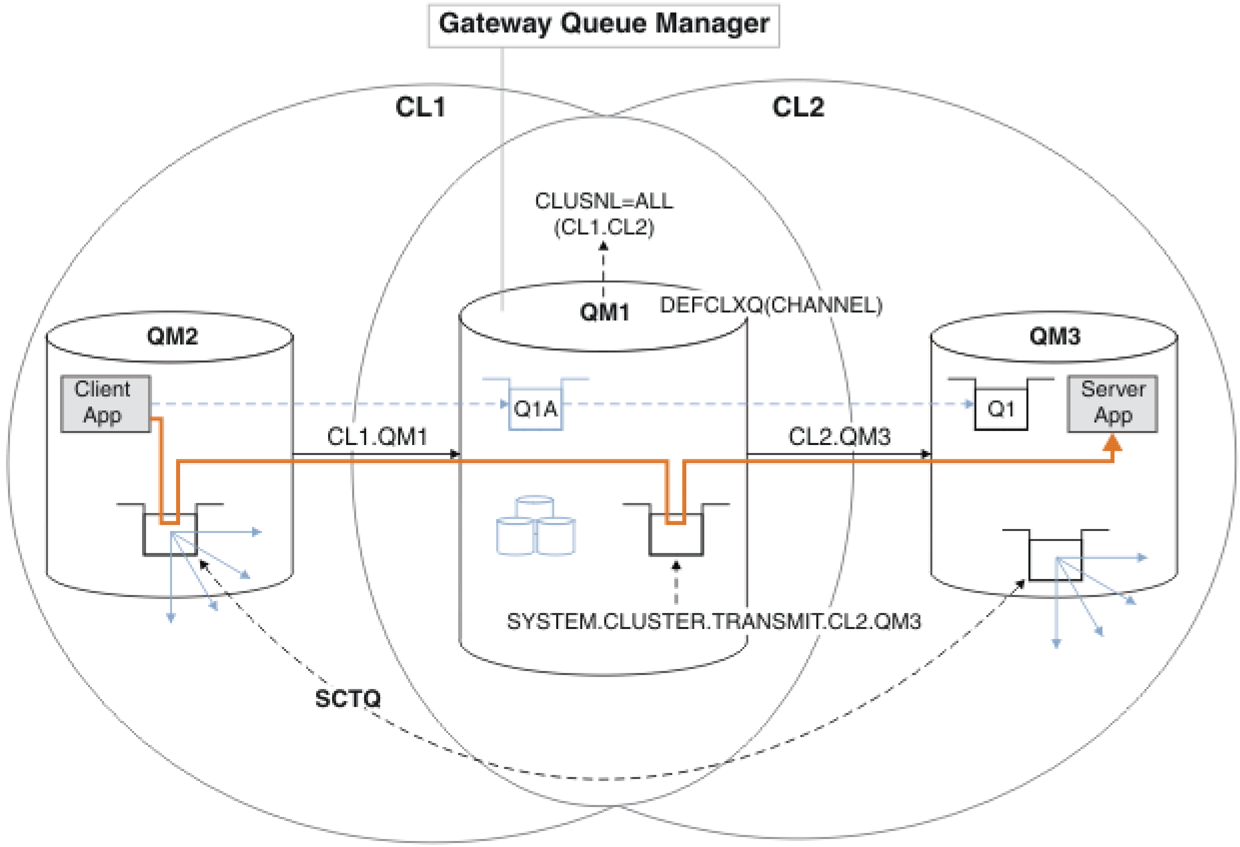

Figure 53. Client-server application deployed to hub and spoke architecture with separate cluster transmission queues on the gateway queue manager.

Procedure

1. Change the gateway queue manager to use separate cluster transmission queues.

```
*... On QM1
ALTER QMGR DEFCLXQ(CHANNEL)
```

2. Switch to the separate cluster transmission queues.

   Any cluster-sender channel that is not running switches to using separate cluster transmission queues when it next starts.

   To switch the running channels, either restart the queue manager, or follow these steps:

   a) List the cluster-sender channels that are running with SYSTEM.CLUSTER.TRANSMIT.QUEUE.

```
*... On QM1
DISPLAY CHSTATUS(*) WHERE(XMITQ EQ 'SYSTEM.CLUSTER.TRANSMIT.QUEUE')
```

   The response is list of channel status reports:

```
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM2)                CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1412))        CURRENT
RQMNAME(QM2)                    STATUS(RUNNING)
SUBSTATE(MQGET)                 XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)                CHLTYPE(CLUSSDR)
```

```
CONNAME(127.0.0.1(1413))    CURRENT
RQMNAME(QM3)                STATUS(RUNNING)
SUBSTATE(MQGET)             XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM5)            CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1415))    CURRENT
RQMNAME(QM5)                STATUS(RUNNING)
SUBSTATE(MQGET)             XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM4)            CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1414))    CURRENT
RQMNAME(QM4)                STATUS(RUNNING)
SUBSTATE(MQGET)             XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
```

b) Stop the channels that are running

For each channel in the list, run the command:

```
*... On QM1
STOP CHANNEL(ChannelName)
```

Where *ChannelName* is each of `CL1.QM2, CL1.QM4, CL1.QM3, CL1.QM5`.

The response is that the command is accepted:

```
AMQ8019: Stop IBM MQ channel accepted.
```

c) Monitor which channels are stopped

```
*... On QM1
DISPLAY CHSTATUS(*) WHERE(XMITQ EQ 'SYSTEM.CLUSTER.TRANSMIT.QUEUE')
```

The response is a list of channels that are still running and channels that are stopped:

```
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM2)           CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1412))   CURRENT
RQMNAME(QM2)               STATUS(STOPPED)
SUBSTATE( )                XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)           CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))   CURRENT
RQMNAME(QM3)               STATUS(STOPPED)
SUBSTATE( )                XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM5)           CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1415))   CURRENT
RQMNAME(QM5)               STATUS(STOPPED)
SUBSTATE( )                XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM4)           CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1414))   CURRENT
RQMNAME(QM4)               STATUS(STOPPED)
SUBSTATE( )                XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
```

d) Start each stopped channel.

Do this step for all the channels that were running. If a channel does not stop, you can run the **STOP CHANNEL** command again with the FORCE option. An example of setting the FORCE option would be if the channel does not stop, and you cannot restart the other queue manager to synchronize the channel.

```
*... On QM1
START CHANNEL(CL2.QM5)
```

The response is that the command is accepted:

```
AMQ8018: Start IBM MQ channel accepted.
```

e) Monitor the transmission queues being switched.

Monitor the gateway queue manager error log for the message " AMQ7341 The `transmission queue for channel CL2.QM3 is SYSTEM.CLUSTER.TRANSMIT.` *`QUEUE|CL2.QM3`* ".

f) Check that `SYSTEM.CLUSTER.TRANSMIT.QUEUE` is no longer used

```
*... On QM1
DISPLAY CHSTATUS(*) WHERE(XMITQ EQ 'SYSTEM.CLUSTER.TRANSMIT.QUEUE')
DISPLAY QUEUE(SYSTEM.CLUSTER.TRANSMIT.QUEUE) CURDEPTH
```

The response is list of channel status reports, and the depth of `SYSTEM.CLUSTER.TRANSMIT.QUEUE`:

```
AMQ8420: Channel Status not found.
AMQ8409: Display Queue details.
QUEUE(SYSTEM.CLUSTER.TRANSMIT.QUEUE)    TYPE(QLOCAL)
CURDEPTH(0)
```

g) Monitor which channels are started

```
*... On QM1
DISPLAY CHSTATUS(*) WHERE(XMITQ LK 'SYSTEM.CLUSTER.TRANSMIT.*')
```

The response is a list of the channels, in this case already running with the new default cluster transmission queues:

```
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM2)                        CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1412))                CURRENT
RQMNAME(QM2)                            STATUS(RUNNING)
SUBSTATE(MQGET)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL1.QM2)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM3)                        CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1413))                CURRENT
RQMNAME(QM3)                            STATUS(RUNNING)
SUBSTATE(MQGET)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL2.QM3)
AMQ8417: Display Channel Status details.
CHANNEL(CL2.QM5)                        CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1415))                CURRENT
RQMNAME(QM5)                            STATUS(RUNNING)
SUBSTATE(MQGET)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL2.QM5)
AMQ8417: Display Channel Status details.
CHANNEL(CL1.QM4)                        CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1(1414))                CURRENT
```

```
RQMNAME(QM4)                            STATUS(RUNNING)
SUBSTATE(MQGET)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL1.QM4)
```

What to do next

1. Test the automatically defined cluster transmission queue by sending a message from QM2 to Q1 on QM3, resolving queue name with the queue alias definition Q1A
   a. Run the sample program **amqsput** on QM2 to put a message.

   ```
   C:\IBM\MQ>amqsput Q1A QM2
   Sample AMQSPUT0 start
   target queue is Q1A
   Sample request message from QM2 to Q1 using Q1A

   Sample AMQSPUT0 end
   ```
   b. Run the sample program **amqsget** to get the message from Q1 on QM3

   ```
   C:\IBM\MQ>amqsget Q1 QM3
   Sample AMQSGET0 start
   message <Sample request message from QM2 to Q1 using Q1A>
   no more messages
   Sample AMQSGET0 end
   ```
2. Consider whether to reconfigure security, by configuring security for the cluster queues on the queue managers where messages for the cluster queues originate.

Related concepts

Access control and multiple cluster transmission queues

Clustering: Application isolation using multiple cluster transmission queues

Related tasks

Adding a remote queue definition to isolate messages sent from a gateway queue manager
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses a clustered queue remote definition, and a separate sender channel and transmission queue.

Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses an additional cluster transmission queue to separate message traffic to a single queue manager in a cluster.

Adding a cluster and a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager
Modify the configuration of overlapping clusters that use a gateway queue manager. After the modification messages are transferred to an application from the gateway queue manager without using the same transmission queue or channels as other cluster messages. The solution uses an additional cluster to isolate the messages to a particular cluster queue.

Clustering: Planning how to configure cluster transmission queues

“Adding a queue manager to a cluster: separate transmission queues” on page 240

Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using multiple cluster transmission queues.

Removing a cluster queue from a queue manager

Disable the INVENTQ queue at Toronto. Send all the inventory messages to New York, and delete the INVENTQ queue at Toronto when it is empty.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The INVENTORY cluster has been set up as described in "Adding a queue manager that hosts a queue" on page 245. It contains four queue managers. LONDON and NEWYORK both hold full repositories. PARIS and TORONTO hold partial repositories. The inventory application runs on the systems in New York and Toronto and is driven by the arrival of messages on the INVENTQ queue.
- Because of reduced workload, you no longer want to run the inventory application in Toronto. You want to disable the INVENTQ queue hosted by the queue manager TORONTO, and have TORONTO feed messages to the INVENTQ queue in NEWYORK.
- Network connectivity exists between all four systems.
- The network protocol is TCP.

About this task

Follow these steps to remove a cluster queue.

Procedure

1. Indicate that the queue is no longer available.

   To remove a queue from a cluster, remove the cluster name from the local queue definition. Alter the INVENTQ on TORONTO so that it is not accessible from the rest of the cluster:

   ```
   ALTER QLOCAL(INVENTQ) CLUSTER(' ')
   ```

2. Check that the queue is no longer available.

   On a full repository queue manager, either LONDON or NEWYORK, check that the queue is no longer hosted by queue manager TORONTO by issuing the following command:

   ```
   DIS QCLUSTER (INVENTQ)
   ```

   TORONTO is not listed in the results, if the ALTER command has completed successfully.

3. Disable the queue.

   Disable the INVENTQ queue at TORONTO so that no further messages can be written to it:

   ```
   ALTER QLOCAL(INVENTQ) PUT(DISABLED)
   ```

   Now messages in transit to this queue using MQOO_BIND_ON_OPEN go to the dead-letter queue. You need to stop all applications from putting messages explicitly to the queue on this queue manager.

4. Monitor the queue until it is empty.

   Monitor the queue using the DISPLAY QUEUE command, specifying the attributes IPPROCS, OPPROCS, and CURDEPTH, or use the **WRKMQMSTS** command on IBM i. When the number of input and output processes, and the current depth of the queueare all zero, the queue is empty.

5. Monitor the channel to ensure there are no in-doubt messages.

   To be sure that there are no in-doubt messages on the channel INVENTORY.TORONTO, monitor the cluster-sender channel called INVENTORY.TORONTO on each of the other queue managers. Issue the DISPLAY CHSTATUS command specifying the INDOUBT parameter from each queue manager:

   ```
   DISPLAY CHSTATUS(INVENTORY.TORONTO) INDOUBT
   ```

   If there are any in-doubt messages, you must resolve them before proceeding. For example, you might try issuing the RESOLVE channel command or stopping and restarting the channel.
6. Delete the local queue.

   When you are satisfied that there are no more messages to be delivered to the inventory application at TORONTO, you can delete the queue:

   ```
   DELETE QLOCAL(INVENTQ)
   ```
7. You can now remove the inventory application from the system in Toronto

   Removing the application avoids duplication and saves space on the system.

Results

The cluster set up by this task is like that setup by the previous task. The difference is the INVENTQ queue is no longer available at queue manager TORONTO.

When you took the queue out of service in step 1, the TORONTO queue manager sent a message to the two full repository queue managers. It notified them of the change in status. The full repository queue managers pass on this information to other queue managers in the cluster that have requested updates to information concerning the INVENTQ.

When a queue manager puts a message on the INVENTQ queue the updated partial repository indicates that the INVENTQ queue is available only at the NEWYORK queue manager. The message is sent to the NEWYORK queue manager.

What to do next

In this task, there was only one queue to remove and only one cluster to remove it from.

Suppose that there are many queues referring to a namelist containing many cluster names. For example, the TORONTO queue manager might host not only the INVENTQ, but also the PAYROLLQ, SALESQ, and PURCHASESQ. TORONTO makes these queues available in all the appropriate clusters, INVENTORY, PAYROLL, SALES, and PURCHASES. Define a namelist of the cluster names on the TORONTO queue manager:

```
DEFINE NAMELIST(TOROLIST)
DESCR('List of clusters TORONTO is in')
NAMES(INVENTORY, PAYROLL, SALES, PURCHASES)
```

Add the namelist to each queue definition:

```
DEFINE QLOCAL(INVENTQ) CLUSNL(TOROLIST)
DEFINE QLOCAL(PAYROLLQ) CLUSNL(TOROLIST)
DEFINE QLOCAL(SALESQ) CLUSNL(TOROLIST)
DEFINE QLOCAL(PURCHASESQ) CLUSNL(TOROLIST)
```

Now suppose that you want to remove all those queues from the SALES cluster, because the SALES operation is to be taken over by the PURCHASES operation. All you need to do is alter the TOROLIST namelist to remove the name of the SALES cluster from it.

If you want to remove a single queue from one of the clusters in the namelist, create a namelist, containing the remaining list of cluster names. Then alter the queue definition to use the new namelist. To remove the PAYROLLQ from the INVENTORY cluster:

1. Create a namelist:

```
DEFINE NAMELIST(TOROSHORTLIST)
DESCR('List of clusters TORONTO is in other than INVENTORY')
NAMES(PAYROLL, SALES, PURCHASES)
```

2. Alter the PAYROLLQ queue definition:

```
ALTER QLOCAL(PAYROLLQ) CLUSNL(TOROSHORTLIST)
```

Removing a queue manager from a cluster

Remove a queue manager from a cluster, in scenarios where the queue manager can communicate normally with at least one full repository in the cluster.

Before you begin

This method is the best practice for scenarios in which at least one full repository is available, and can be contacted by the queue manager that is being removed. This method involves the least manual intervention, and allows the queue manager to negotiate a controlled withdrawal from the cluster. If the queue manager that is being removed cannot contact a full repository, see "Removing a queue manager from a cluster: Alternative method" on page 287.

About this task

This example task removes the queue manager LONDON from the INVENTORY cluster. The INVENTORY cluster is set up as described in "Adding a queue manager to a cluster" on page 237, and modified as described in "Removing a cluster queue from a queue manager" on page 283.

The process for removing a queue manager from a cluster is more complicated than the process of adding a queue manager.

When a queue manager joins a cluster, the existing members of the cluster have no knowledge of the new queue manager and so have no interactions with it. New sender and receiver channels must be created on the joining queue manager so that it can connect to a full repository.

When a queue manager is removed from a cluster, it is likely that applications connected to the queue manager are using objects such as queues that are hosted elsewhere in the cluster. Also, applications that are connected to other queue managers in the cluster might be using objects hosted on the target queue manager. As a result of these applications, the current queue manager might create additional sender channels to establish communication with cluster members other than the full repository that it used to join the cluster. Every queue manager in the cluster has a cached copy of data that describes other cluster members. This might include the one that is being removed.

Procedure

1. Before you remove the queue manager from the cluster, ensure that the queue manager is no longer hosting resources that are needed by the cluster:
   - If the queue manager hosts a full repository, complete steps 1-6 from "Moving a full repository to another queue manager" on page 249. If the full repository functionality of the queue manager to be removed is not to be moved to a different queue manager, it is only necessary to complete steps 5 and 6.
   - If the queue manager hosts cluster queues, complete steps 1-7 from "Removing a cluster queue from a queue manager" on page 283.

- If the queue manager hosts cluster topics, either delete the topics (for example by using the DELETE TOPIC command), or move them to other hosts as described in "Moving a cluster topic definition to a different queue manager" on page 338.

  Note: If you remove a queue manager from a cluster, and the queue manager still hosts a cluster topic, then the queue manager might continue to attempt to deliver publications to the queue managers that are left in the cluster until the topic is deleted.

2. Alter the manually defined cluster receiver channels to remove them from the cluster, on queue manager LONDON:

   ```
   ALTER CHANNEL(INVENTORY.LONDON) CHLTYPE(CLUSRCVR) CLUSTER(' ')
   ```

3. Alter the manually defined cluster sender channels to remove them from the cluster, on queue manager LONDON:

   ```
   ALTER CHANNEL(INVENTORY.PARIS) CHLTYPE(CLUSSDR) CLUSTER(' ')
   ```

   The other queue managers in the cluster learn that this queue manager and its cluster resources are no longer part of the cluster.

4. Monitor the cluster transmit queue, on queue manager LONDON, until there are no messages that are waiting to flow to any full repository in the cluster.

   ```
   DISPLAY CHSTATUS(INVENTORY.LONDON) XQMSGSA
   ```

   If messages remain on the transmit queue, determine why they are not being sent to the PARIS and NEWYORK full repositories before continuing.

Results

The queue manager LONDON is no longer part of the cluster. However, it can still function as an independent queue manager.

What to do next

The result of these changes can be confirmed by issuing the following command on the remaining members of the cluster:

```
DISPLAY CLUSQMGR(LONDON)
```

The queue manager continues to be displayed until the auto-defined cluster sender channels to it have stopped. You can wait for this to happen, or, continue to monitor for active instances by issuing the following command:

```
DISPLAY CHANNEL(INVENTORY.LONDON)
```

When you are confident that no more messages are being delivered to this queue manager, you can stop the cluster sender channels to LONDON by issuing the following command on the remaining members of the cluster:

```
STOP CHANNEL(INVENTORY.LONDON) STATUS(INACTIVE)
```

After the changes are propagated throughout the cluster, and no more messages are being delivered to this queue manager, stop and delete the CLUSRCVR channel on LONDON:

```
STOP CHANNEL(INVENTORY.LONDON)
DELETE CHANNEL(INVENTORY.LONDON)
```

The removed queue manager can be added back into the cluster at a later point as described in "Adding a queue manager to a cluster" on page 237. The removed queue manager continues to cache knowledge of the remaining members of the cluster for up to 90 days. If you prefer not to wait until this cache expires, it can be forcibly removed as described in "Restoring a queue manager to its pre-cluster state" on page 289.

Removing a queue manager from a cluster: Alternative method

Remove a queue manager from a cluster, in scenarios where, because of a significant system or configuration issue, the queue manager cannot communicate with any full repository in the cluster.

Before you begin

This alternative method of removing a queue manager from a cluster manually stops and deletes all cluster channels linking the removed queue manager to the cluster, and forcibly removes the queue manager from the cluster. This method is used in scenarios where the queue manager that is being removed cannot communicate with any of the full repositories. This might be (for example) because the queue manager has stopped working, or because there has been a prolonged communications failure between the queue manager and the cluster. Otherwise, use the most common method: "Removing a queue manager from a cluster" on page 285.

About this task

This example task removes the queue manager LONDON from the INVENTORY cluster. The INVENTORY cluster is set up as described in "Adding a queue manager to a cluster" on page 237, and modified as described in "Removing a cluster queue from a queue manager" on page 283.

The process for removing a queue manager from a cluster is more complicated than the process of adding a queue manager.

When a queue manager joins a cluster, the existing members of the cluster have no knowledge of the new queue manager and so have no interactions with it. New sender and receiver channels must be created on the joining queue manager so that it can connect to a full repository.

When a queue manager is removed from a cluster, it is likely that applications connected to the queue manager are using objects such as queues that are hosted elsewhere in the cluster. Also, applications that are connected to other queue managers in the cluster might be using objects hosted on the target queue manager. As a result of these applications, the current queue manager might create additional sender channels to establish communication with cluster members other than the full repository that it used to join the cluster. Every queue manager in the cluster has a cached copy of data that describes other cluster members. This might include the one that is being removed.

This procedure might be appropriate in an emergency, when it is not possible to wait for the queue manager to leave the cluster gracefully.

Procedure

1. Before you remove the queue manager from the cluster, ensure that the queue manager is no longer hosting resources that are needed by the cluster:
   - If the queue manager hosts a full repository, complete steps 1-6 from "Moving a full repository to another queue manager" on page 249. If the full repository functionality of the queue manager to be removed is not to be moved to a different queue manager, it is only necessary to complete steps 5 and 6.

- If the queue manager hosts cluster queues, complete steps 1-7 from "Removing a cluster queue from a queue manager" on page 283.
- If the queue manager hosts cluster topics, either delete the topics (for example by using the DELETE TOPIC command), or move them to other hosts as described in "Moving a cluster topic definition to a different queue manager" on page 338.

  Note: If you remove a queue manager from a cluster, and the queue manager still hosts a cluster topic, then the queue manager might continue to attempt to deliver publications to the queue managers that are left in the cluster until the topic is deleted.

2. Stop all channels used to communicate with other queue managers in the cluster. Use `MODE(FORCE)` to stop the `CLUSRCVR` channel, on queue manager `LONDON`. Otherwise you might need to wait for the sender queue manager to stop the channel:

```
STOP CHANNEL(INVENTORY.LONDON) MODE(FORCE)
STOP CHANNEL(INVENTORY.TORONTO)
STOP CHANNEL(INVENTORY.PARIS)
STOP CHANNEL(INVENTORY.NEWYORK)
```

3. Monitor the channel states, on queue manager `LONDON`, until the channels stop:

```
DISPLAY CHSTATUS(INVENTORY.LONDON)
DISPLAY CHSTATUS(INVENTORY.TORONTO)
DISPLAY CHSTATUS(INVENTORY.PARIS)
DISPLAY CHSTATUS(INVENTORY.NEWYORK)
```

   No more application messages are sent to or from the other queue managers in the cluster after the channels stop.

4. Delete the manually defined cluster channels, on queue manager `LONDON`:

```
DELETE CHANNEL(INVENTORY.NEWYORK)
DELETE CHANNEL(INVENTORY.TORONTO)
```

5. The remaining queue managers in the cluster still retain knowledge of the removed queue manager, and might continue to send messages to it. To purge the knowledge from the remaining queue managers, reset the removed queue manager from the cluster on one of the full repositories:

```
RESET CLUSTER(INVENTORY) ACTION(FORCEREMOVE) QMNAME(LONDON) QUEUES(YES)
```

   If there might be another queue manager in the cluster that has the same name as the removed queue manager, specify the **`QMID`** of the removed queue manager.

Results

The queue manager `LONDON` is no longer part of the cluster. However, it can still function as an independent queue manager.

What to do next

The result of these changes can be confirmed by issuing the following command on the remaining members of the cluster:

```
DISPLAY CLUSQMGR(LONDON)
```

The queue manager continues to be displayed until the auto-defined cluster sender channels to it have stopped. You can wait for this to happen, or, continue to monitor for active instances by issuing the following command:

```
DISPLAY CHANNEL(INVENTORY.LONDON)
```

After the changes are propagated throughout the cluster, and no more messages are being delivered to this queue manager, delete the CLUSRCVR channel on LONDON:

```
DELETE CHANNEL(INVENTORY.LONDON)
```

The removed queue manager can be added back into the cluster at a later point as described in "Adding a queue manager to a cluster" on page 237. The removed queue manager continues to cache knowledge of the remaining members of the cluster for up to 90 days. If you prefer not to wait until this cache expires, it can be forcibly removed as described in "Restoring a queue manager to its pre-cluster state" on page 289.

Restoring a queue manager to its pre-cluster state

When a queue manager is removed from a cluster, it retains knowledge of the remaining cluster members. This knowledge eventually expires and is deleted automatically. However, if you prefer to delete it immediately, you can use the steps in this topic.

Before you begin

It is assumed that the queue manager has been removed from the cluster, and is no longer performing any work in the cluster. For example, its queues are no longer receiving messages from the cluster, and no applications are waiting for messages to arrive in these queues.

About this task

When a queue manager is removed from a cluster, it retains knowledge of the remaining cluster members for up to 90 days. This can have system benefits, particularly if the queue manager quickly rejoins the cluster. When this knowledge eventually expires, it is deleted automatically. However, there are reasons why you might prefer to delete this information manually. For example:

- You might want to confirm that you have stopped every application on this queue manager that previously used cluster resources. Until the knowledge of the remaining cluster members expires, any such application continues to write to a transmit queue. After the cluster knowledge is deleted, the system generates an error message when such an application tries to use cluster resources.
- When you display status information for the queue manager, you might prefer not to see expiring information about remaining cluster members.

This task uses the INVENTORY cluster as an example. The LONDON queue manager has been removed from the INVENTORY cluster as described in "Removing a queue manager from a cluster" on page 285. To delete knowledge of the remaining members of the cluster, issue the following commands on the LONDON queue manager.

Procedure

1. Remove all memory of the other queue managers in the cluster from this queue manager:

   ```
   REFRESH CLUSTER(INVENTORY) REPOS(YES)
   ```

2. Monitor the queue manager until all the cluster resources are gone:

   ```
   DISPLAY CLUSQMGR(*) CLUSTER(INVENTORY)
   DISPLAY QCLUSTER(*) CLUSTER(INVENTORY)
   DISPLAY TOPIC(*) CLUSTER(INVENTORY)
   ```

Related concepts
Clusters
Comparison of clustering and distributed queuing
Cluster components

Maintaining a queue manager

Suspend and resume a queue manager from a cluster to perform maintenance.

About this task

From time to time, you might need to perform maintenance on a queue manager that is part of a cluster. For example, you might need to take backups of the data in its queues, or apply fixes to the software. If the queue manager hosts any queues, its activities must be suspended. When the maintenance is complete, its activities can be resumed.

Procedure

1. Suspend a queue manager, by issuing the SUSPEND QMGR **runmqsc** command:

   ```
   SUSPEND QMGR CLUSTER(SALES)
   ```

   The SUSPEND **runmqsc** command notifies the queue managers in the SALES cluster that this queue manager has been suspended.

   The purpose of the SUSPEND QMGR command is only to advise other queue managers to avoid sending messages to this queue manager if possible. It does not mean that the queue manager is disabled. Some messages that have to be handled by this queue manager are still sent to it, for example when this queue manager is the only host of a clustered queue.

   While the queue manager is suspended the workload management routines avoid sending messages to it. Messages that have to be handled by that queue manager include messages sent by the local queue manager.

   IBM MQ uses a workload balancing algorithm to determine which destinations are suitable, rather than selecting the local queue manager whenever possible.

   a) Enforce the suspension of a queue manager by using the FORCE option on the SUSPEND QMGR command:

      ```
      SUSPEND QMGR CLUSTER(SALES) MODE(FORCE)
      ```

      MODE(FORCE) forcibly stops all inbound channels from other queue managers in the cluster. If you do not specify MODE(FORCE), the default MODE(QUIESCE) applies.
2. Do whatever maintenance tasks are necessary.
3. Resume the queue manager by issuing the RESUME QMGR **runmqsc** command:

   ```
   RESUME QMGR CLUSTER(SALES)
   ```

Results

The RESUME **runmqsc** command notifies the full repositories that the queue manager is available again. The full repository queue managers disseminate this information to other queue managers that have requested updates to information concerning this queue manager.

Maintaining the cluster transmission queue

Make every effort to keep cluster transmission queues available. They are essential to the performance of clusters. z/OS On z/OS, set the `INDXTYPE` of a cluster transmission queue to `CORRELID`.

Before you begin

- Make sure that the cluster transmission queue does not become full.
- Take care not to issue an `ALTER` **`runmqsc`** command to set it either get disabled or put disabled accidentally.
- Make sure that the medium the cluster transmission queue is stored on z/OS (for example z/OS page sets) does not become full.

About this task

z/OS

The following procedure is only applicable to z/OS.

Procedure

Set the `INDXTYPE` of the cluster transmission queue to `CORRELID`

Refreshing a cluster queue manager

You can remove auto-defined channels and auto-defined cluster objects from the local repository using the `REFRESH CLUSTER` command. No messages are lost.

Before you begin

You might be asked to use the command by your IBM Support Center. Do not use the command without careful consideration. For example, for large clusters use of the **`REFRESH CLUSTER`** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Clustering: Using REFRESH CLUSTER best practices.

About this task

A queue manager can make a fresh start in a cluster. In normal circumstances, you do not need to use the `REFRESH CLUSTER` command.

Procedure

Issue the `REFRESH CLUSTER` **`MQSC`** command from a queue manager to remove auto-defined cluster queue-manager and queue objects from the local repository.

The command only removes objects that refer to other queue managers, it does not remove objects relating to the local queue manager. The command also removes auto-defined channels. It removes channels that do not have messages on the cluster transmission queue and are not attached to a full repository queue manager.

Results

Effectively, the `REFRESH CLUSTER` command allows a queue manager to be cold-started with respect to its full repository content. IBM MQ ensures that no data is lost from your queues.

Related information

Clustering: Using REFRESH CLUSTER best practices

Recovering a cluster queue manager

Bring the cluster information about a queue manager up to date using the REFRESH CLUSTER **runmqsc** command. Follow this procedure after recovering a queue manager from a point-in-time backup.

Before you begin

You have restored a cluster queue manager from a point-in-time backup.

About this task

To recover a queue manager in a cluster, restore the queue manager, and then bring the cluster information up to date using the REFRESH CLUSTER **runmqsc** command.

Note: For large clusters, using the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

Procedure

Issue the REFRESH CLUSTER command on the restored queue manager for all clusters in which the queue manager participates.

What to do next

There is no need to issue the REFRESH CLUSTER command on any other queue manager.

Related information

Clustering: Using REFRESH CLUSTER best practices

Configuring cluster channels for availability

Follow good configuration practices to keep cluster channels running smoothly if there are intermittent network stoppages.

Before you begin

Clusters relieve you of the need to define channels, but you still need to maintain them. The same channel technology is used for communication between queue managers in a cluster as is used in distributed queuing. To understand about cluster channels, you need to be familiar with matters such as:

- How channels operate
- How to find their status
- How to use channel exits

About this task

You might want to give some special consideration to the following points:

Procedure

Consider the following points when configuring cluster channels

- Choose values for HBINT or KAINT on cluster-sender channels and cluster-receiver channels that do not burden the network with lots of heartbeat or keep alive flows. An interval less than about 10 seconds gives false failures, if your network sometimes slows down and introduces delays of this length.
- Set the BATCHHB value to reduce the window for causing a marooned message because it is indoubt on a failed channel. An indoubt batch on a failed channel is more likely to occur if the batch is given

longer to fill. If the message traffic along the channel is sporadic with long periods of time between bursts of messages a failed batch is more likely.

- A problem arises if the cluster-sender end of a channel fails and then tries to restart before the heartbeat or keep alive has detected the failure. The channel-sender restart is rejected if the cluster-receiver end of the channel has remained active. To avoid the failure, arrange for the cluster-receiver channel to be terminated and restarted when a cluster-sender channel attempts to restart.

  z/OS **On IBM MQ for z/OS**
  Control the problem of the cluster-receiver end of the channel remaining active using the `ADOPTMCA` and `ADOPTCHK` parameters on `ALTER QMGR`.

  On platforms other than z/OS
  Control the problem of the cluster-receiver end of the channel remaining active using the `AdoptNewMCA`, `AdoptNewMCATimeout`, and `AdoptNewMCACheck` attributes in the `qm.ini` file or the Windows NT Registry.

Routing messages to and from clusters

Use queue aliases, queue manager aliases, and remote queue definitions to connect clusters to external queue managers and other clusters.

For details on routing messages to and from clusters, see the following subtopics:

Related concepts

Clusters

Comparison of clustering and distributed queuing

Components of a cluster

“Queue-manager aliases and clusters” on page 307
Use queue-manager aliases to hide the name of queue managers when sending messages into or out of a cluster, and to workload balance messages sent to a cluster.

“Queue aliases and clusters” on page 310
Use queue aliases to hide the name of a cluster queue, to cluster a queue, adopt different attributes, or adopt different access controls.

“Reply-to queue aliases and clusters” on page 310
A reply-to queue alias definition is used to specify alternative names for reply information. Reply-to queue alias definitions can be used with clusters just the same as in a distributed queuing environment.

Related tasks

“Configuring a queue manager cluster” on page 215
Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. You can define cluster components, and create and manage clusters.

“Setting up a new cluster” on page 227

Follow these instructions to set up the example cluster. Separate instructions describe setting up the cluster on TCP/IP, LU 6.2, and with a single transmission queue or multiple transmission queues. Test the cluster works by sending a message from one queue manager to the other.

Configuring request/reply to a cluster

Configure a request/reply message path from a queue manager outside a cluster. Hide the inner details of the cluster by using a gateway queue manager as the communication path to and from the cluster.

Before you begin

Figure 54 on page 295 shows a queue manager called QM3 that is outside the cluster called DEMO. QM3 could be a queue manager on an IBM MQ product that does not support clusters. QM3 hosts a queue called Q3, which is defined as follows:

```
DEFINE QLOCAL(Q3)
```

Inside the cluster are two queue managers called QM1 and QM2. QM2 hosts a cluster queue called Q2, which is defined as follows:

```
DEFINE QLOCAL(Q2) CLUSTER(DEMO)
```

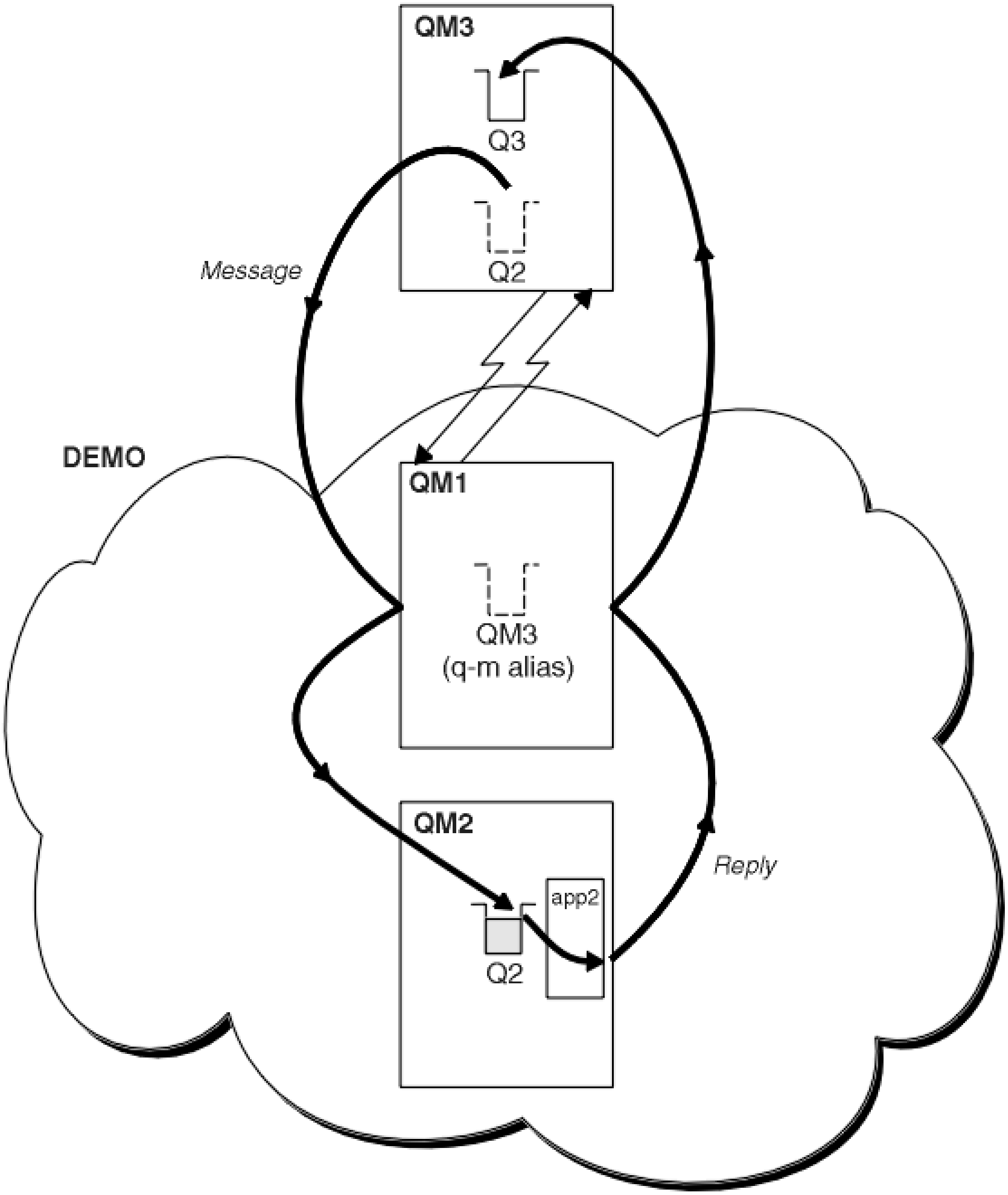

Figure 54. Putting from a queue manager outside the cluster

About this task

Follow the advice in the procedure to set up the path for the request and reply messages.

Procedure

1. Send the request message to the cluster.

   Consider how the queue manager that is outside the cluster puts a message to the queue `Q2` at `QM2`, that is inside the cluster. A queue manager outside the cluster must have a `QREMOTE` definition for each queue in the cluster that it puts messages to.

   a) Define a remote queue for `Q2` on `QM3`.

```
DEFINE QREMOTE(Q2) RNAME(Q2) RQMNAME(QM2) XMITQ(QM1)
```

Because QM3 is not part of a cluster, it must communicate using distributed queuing techniques. Therefore, it must also have a sender-channel and a transmission queue to QM1. QM1 needs a corresponding receiver channel. The channels and transmission queues are not shown explicitly in Figure 54 on page 295.

In the example, an application at QM3 issues an MQPUT call to put a message to Q2. The QREMOTE definition causes the message to be routed to Q2 at QM2 using the sender-channel that is getting messages from the QM1 transmission queue.

2. Receive the reply message from the cluster.

   Use a queue manager alias to create a return path for replies to a queue manager outside the cluster. The gateway, QM1, advertises a queue-manager alias for the queue manager that is outside the cluster, QM3. It advertises QM3 to the queue managers inside the cluster by adding the cluster attribute to a queue manager alias definition for QM3. A queue manager alias definition is like a remote queue definition, but with a blank RNAME.

   a) Define a queue manager alias for QM3 on QM1.

   ```
   DEFINE QREMOTE(QM3) RNAME(' ') RQMNAME(QM3) CLUSTER(DEMO)
   ```

   We must consider the choice of name for the transmission queue used to forward replies back from QM1 to QM3. Implicit in the QREMOTE definition, by the omission of the XMITQ attribute, is the name of the transmission queue is QM3. But QM3 is the same name as we expect to advertise to the rest of the cluster using the queue manager alias. IBM MQ does not allow you to give both the transmission queue and the queue manager alias the same name. One solution is to create a transmission queue to forward messages to QM3 with a different name to the queue manager alias.

   b) Provide the transmission queue name in the QREMOTE definition.

   ```
   DEFINE QREMOTE(QM3) RNAME(' ') RQMNAME(QM3) CLUSTER(DEMO) XMITQ(QM3.XMIT)
   ```

   The new queue manager alias couples the new transmission queue called QM3.XMIT with the QM3 queue manager alias. It is a simple and correct solution, but not wholly satisfactory. It has broken the naming convention for transmission queues that they are given the same name as the target queue manager. Are there any alternative solutions that preserve the transmission queue naming convention?

   The problem arises because the requester defaulted to passing QM3 as the reply-to queue manager name in the request message that is sent from QM3. The server on QM2 uses the QM3 reply-to queue manager name to address QM3 in its replies. The solution required QM1 to advertise QM3 as the queue manager alias to return reply messages to and prevented QM1 from using QM3 as the name of the transmission queue.

   Instead of defaulting to providing QM3 as the reply-to queue manager name, applications on QM3 need to pass a reply-to queue manager alias to QM1 for reply messages. The gateway queue manager QM1 advertises the queue manager alias for replies to QM3 rather than QM3 itself, avoiding the conflict with the name of the transmission queue.

   c) Define a queue manager alias for QM3 on QM1.

   ```
   DEFINE QREMOTE(QM3.ALIAS) RNAME(' ') RQMNAME(QM3) CLUSTER(DEMO)
   ```

   Two changes to the configuration commands are required.

   i) The QREMOTE at QM1 now advertises our queue manager alias QM3.ALIAS to the rest of the cluster, coupling it to the name of the real queue manager QM3. QM3 is again the name of the transmission queue to send reply queues back to QM3

ii) The client application must provide QM3.ALIAS as the name of the reply-to queue manager when it constructs the request message. You can provide QM3.ALIAS to the client application in one of two ways.

- Code QM3.ALIAS in the reply-to queue manager name field constructed by MQPUT in the MQMD. You must do it this way if you are using a dynamic queue for replies.
- Use a reply-to queue alias, Q3.ALIAS, rather than a reply-to queue when providing the reply-to queue name.

```
DEFINE QREMOTE(Q3.ALIAS) RNAME(Q3) RQMNAME(QM3.ALIAS)
```

What to do next

Note: You cannot demonstrate the use of reply-to queue aliases with **AMQSREQ0**. It opens the reply-to queue using the queue name provided in parameter 3, or the default SYSTEM.SAMPLE.REPLY model queue. You need to modify the sample providing another parameter containing the reply-to queue alias to name the reply-to queue manager alias for MQPUT.

Related concepts

Queue-manager aliases and clusters
Use queue-manager aliases to hide the name of queue managers when sending messages into or out of a cluster, and to workload balance messages sent to a cluster.

Reply-to queue aliases and clusters
A reply-to queue alias definition is used to specify alternative names for reply information. Reply-to queue alias definitions can be used with clusters just the same as in a distributed queuing environment.

Queue aliases and clusters
Use queue aliases to hide the name of a cluster queue, to cluster a queue, adopt different attributes, or adopt different access controls.

Related tasks

Configuring request/reply from a cluster
Configure a request/reply message path from a cluster to a queue manager outside the cluster. Hide the details of how a queue manager inside the cluster communicates outside the cluster by using a gateway queue manager.

Configuring workload balancing from outside a cluster
Configure a message path from a queue manager outside a cluster to any copy of a cluster queue. The result is to workload balance requests from outside the cluster to each instance of a cluster queue.

Configuring message paths between clusters
Connect clusters together using a gateway queue manager. Make queues or queue managers visible to all the clusters by defining cluster queue or cluster queue manager aliases on the gateway queue manager.

"Hiding the name of a cluster target queue manager" on page 297
Route a message to a cluster queue that is defined on any queue manager in a cluster without naming the queue manager.

Hiding the name of a cluster target queue manager
Route a message to a cluster queue that is defined on any queue manager in a cluster without naming the queue manager.

Before you begin

- Avoid revealing the names of queue managers that are inside the cluster to queue managers that are outside the cluster.
  - Resolving references to the queue manager hosting a queue inside the cluster removes the flexibility to do workload balancing.
  - It also makes it difficult for you to change a queue manager hosting a queue in the cluster.

- The alternative is to replace RQMNAME with a queue manager alias provided by the cluster administrator.
- "Hiding the name of a cluster target queue manager" on page 297 describes using a queue manager alias to decouple a queue manager outside a cluster from the management of queue managers inside a cluster.

- However, the suggested way to name transmission queues is to give them the name of the target queue manager. The name of the transmission queue reveals the name of a queue manager in the cluster. You have to choose which rule to follow. You might choose to name the transmission queue using either the queue manager name or the cluster name:

  Name the transmission queue using the gateway queue manager name
  Disclosure of the gateway queue manager name to queue managers outside a cluster is a reasonable exception to the rule of hiding cluster queue manager names.

  Name the transmission queue using the name of the cluster
  If you are not following the convention of naming transmission queues with the name of the target queue manager, use the cluster name.

About this task

Modify the task "Configuring request/reply to a cluster" on page 294, to hide the name of the target queue manager inside the cluster.

Procedure

In the example, see Figure 55 on page 299, define a queue manager alias on the gateway queue manager QM1 called DEMO:

```
DEFINE QREMOTE(DEMO) RNAME(' ') RQMNAME(' ')
```

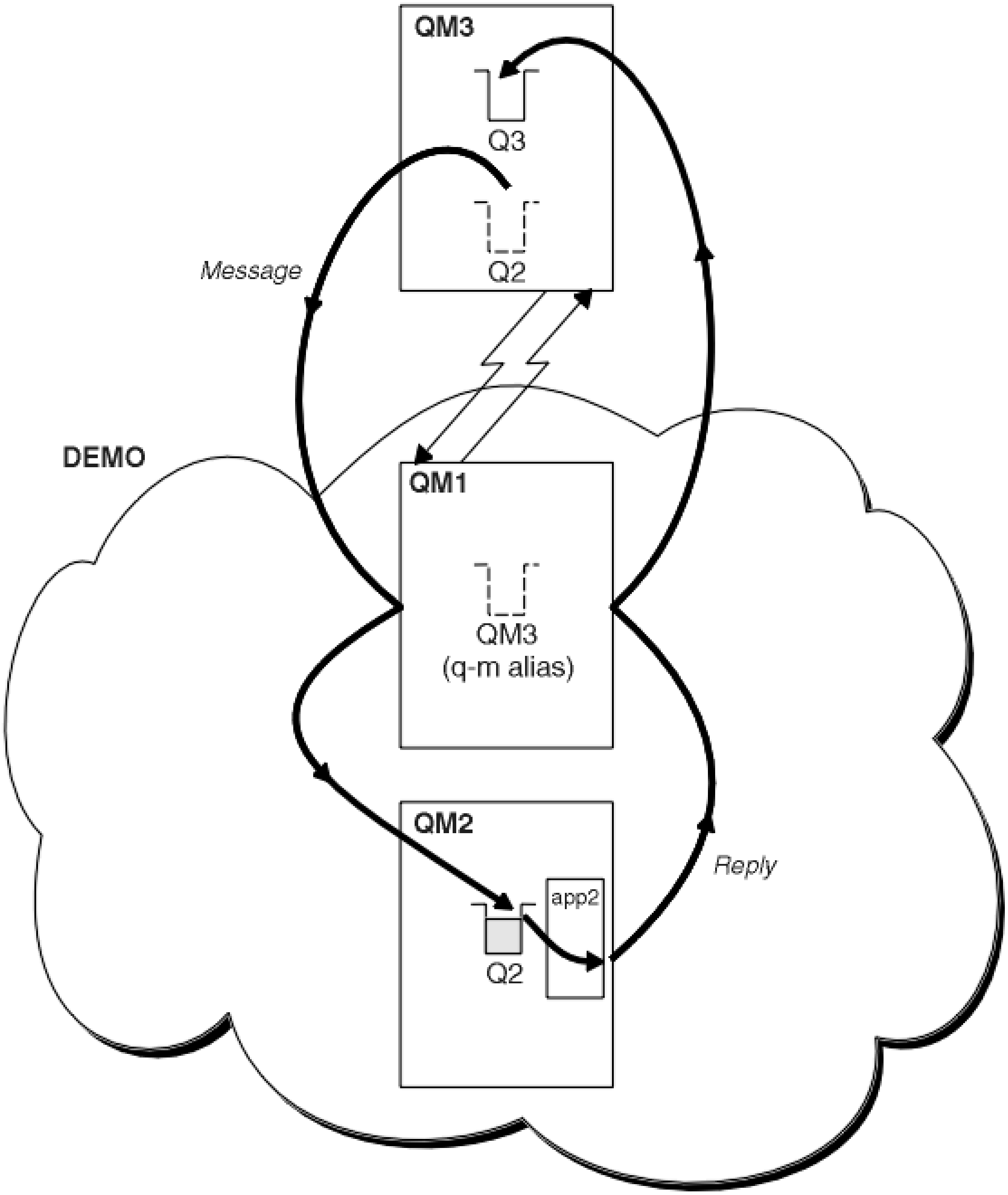

Figure 55. Putting from a queue manager outside the cluster

The QREMOTE definition on QM1 makes the queue manager alias DEMO known to the gateway queue manager. QM3, the queue manager outside the cluster, can use the queue manager alias DEMO to send messages to cluster queues on DEMO, rather than having to use an actual queue manager name.

If you adopt the convention of using the cluster name to name the transmission queue connecting to a cluster, then the remote queue definition for Q2 becomes:

```
DEFINE QREMOTE(Q2) RNAME(Q2) RQMNAME(DEMO) XMIT(DEMO)
```

Results

Messages destined for Q2 on DEMO are placed on the DEMO transmission queue. From the transmission queue they are transferred by the sender-channel to the gateway queue manager, QM1. The gateway queue manager routes the messages to any queue manager in the cluster that hosts the cluster queue Q2.

Configuring request/reply from a cluster

Configure a request/reply message path from a cluster to a queue manager outside the cluster. Hide the details of how a queue manager inside the cluster communicates outside the cluster by using a gateway queue manager.

Before you begin

Figure 56 on page 301 shows a queue manager, QM2, inside the cluster DEMO. It sends a request to a queue, Q3, hosted on queue manager outside the cluster. The replies are returned to Q2 at QM2 inside the cluster.

To communicate with the queue manager outside the cluster, one or more queue managers inside the cluster act as a gateway. A gateway queue manager has a communication path to the queue managers outside the cluster. In the example, QM1 is the gateway.

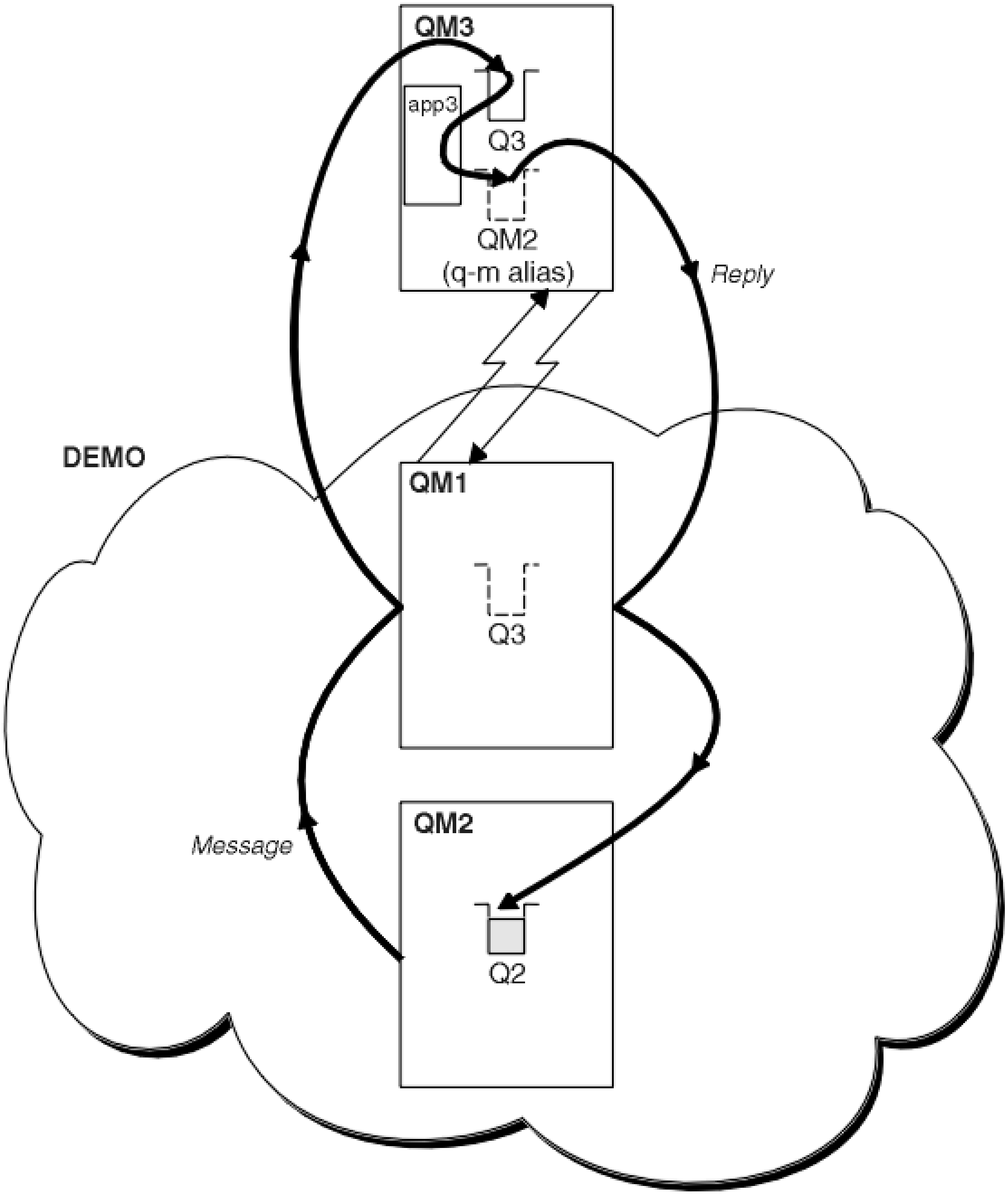

Figure 56. Putting to a queue manager outside the cluster

About this task

Follow the instructions to set up the path for the request and reply messages

Procedure

1. Send the request message from the cluster.

   Consider how the queue manager, QM2, which is inside the cluster puts a message to the queue Q3 at QM3, which is outside the cluster.

   a) Create a QREMOTE definition on QM1 that advertises the remote queue Q3 to the cluster

   ```
   DEFINE QREMOTE(Q3) RNAME(Q3) RQMNAME(QM3) CLUSTER(DEMO)
   ```

It also has a sender-channel and a transmission queue to the queue manager that is outside the cluster. QM3 has a corresponding receiver-channel. The channels are not shown in Figure 56 on page 301.

An application on QM2 issues an MQPUT call specifying the target queue and the queue to which replies are to be sent. The target queue is Q3 and the reply-to queue is Q2.

The message is sent to QM1, which uses its remote-queue definition to resolve the queue name to Q3 at QM3.

2. Receive the reply message from the queue manager outside the cluster.

   A queue manager outside the cluster must have a queue manager alias for each queue manager in the cluster to which it send a message. The queue-manager alias must also specify the name of the transmission queue to the gateway queue manager. In this example, QM3 needs a queue manager alias definition for QM2:

   a) Create a queue manager alias QM2 on QM3

   ```
   DEFINE QREMOTE(QM2) RNAME(' ') RQMNAME(QM2) XMITQ(QM1)
   ```

   QM3 also needs a sender-channel and transmission queue to QM1 and QM1 needs a corresponding receiver-channel.

   The application, **app3**, on QM3 can then send replies to QM2, by issuing an MQPUT call and specifying the queue name, Q2 and the queue manager name, QM2.

What to do next

You can define more than one route out of a cluster.

Related concepts

Queue-manager aliases and clusters
Use queue-manager aliases to hide the name of queue managers when sending messages into or out of a cluster, and to workload balance messages sent to a cluster.

Reply-to queue aliases and clusters
A reply-to queue alias definition is used to specify alternative names for reply information. Reply-to queue alias definitions can be used with clusters just the same as in a distributed queuing environment.

Queue aliases and clusters
Use queue aliases to hide the name of a cluster queue, to cluster a queue, adopt different attributes, or adopt different access controls.

Related tasks

Configuring request/reply to a cluster
Configure a request/reply message path from a queue manager outside a cluster. Hide the inner details of the cluster by using a gateway queue manager as the communication path to and from the cluster.

Configuring workload balancing from outside a cluster
Configure a message path from a queue manager outside a cluster to any copy of a cluster queue. The result is to workload balance requests from outside the cluster to each instance of a cluster queue.

Configuring message paths between clusters

Connect clusters together using a gateway queue manager. Make queues or queue managers visible to all the clusters by defining cluster queue or cluster queue manager aliases on the gateway queue manager.

Configuring workload balancing from outside a cluster

Configure a message path from a queue manager outside a cluster to any copy of a cluster queue. The result is to workload balance requests from outside the cluster to each instance of a cluster queue.

Before you begin

Configure the example, as shown in Figure 54 on page 295 in "Configuring request/reply to a cluster" on page 294.

About this task

In this scenario, the queue manager outside the cluster, QM3 in Figure 57 on page 304, sends requests to the queue Q2. Q2 is hosted on two queue managers, QM2 and QM4 within cluster DEMO. Both queue managers are configured with a default bind option of NOTFIXED in order to use workload balancing. The requests from QM3, the queue manager outside the cluster, are sent to either instance of Q2 through QM1.

QM3 is not part of a cluster and communicates using distributed queuing techniques. It must have a sender-channel and a transmission queue to QM1. QM1 needs a corresponding receiver-channel. The channels and transmission queues are not shown explicitly in Figure 57 on page 304.

The procedure extends the example in Figure 54 on page 295 in "Configuring request/reply to a cluster" on page 294.

Procedure

1. Create a QREMOTE definition for Q2 on QM3.

   ```
   DEFINE QREMOTE(Q2) RNAME(Q2) RQMNAME(Q3) XMITQ(QM1)
   ```

   Create a QREMOTE definition for each queue in the cluster that QM3 puts messages to.
2. Create a queue-manager alias Q3 on QM1.

   ```
   DEFINE QREMOTE(Q3) RNAME(' ') RQMNAME(' ')
   ```

   Q3 is not a real queue manager name. It is the name of a queue manager alias definition in the cluster that equates the queue manager alias name Q3 with blank, ' '
3. Define a local queue called Q2 on each of QM2 and QM4.

   ```
   DEFINE QLOCAL(Q2) CLUSTER(DEMO) DEFBIND(NOTFIXED)
   ```
4. QM1, the gateway queue manager, has no special definitions.

Results

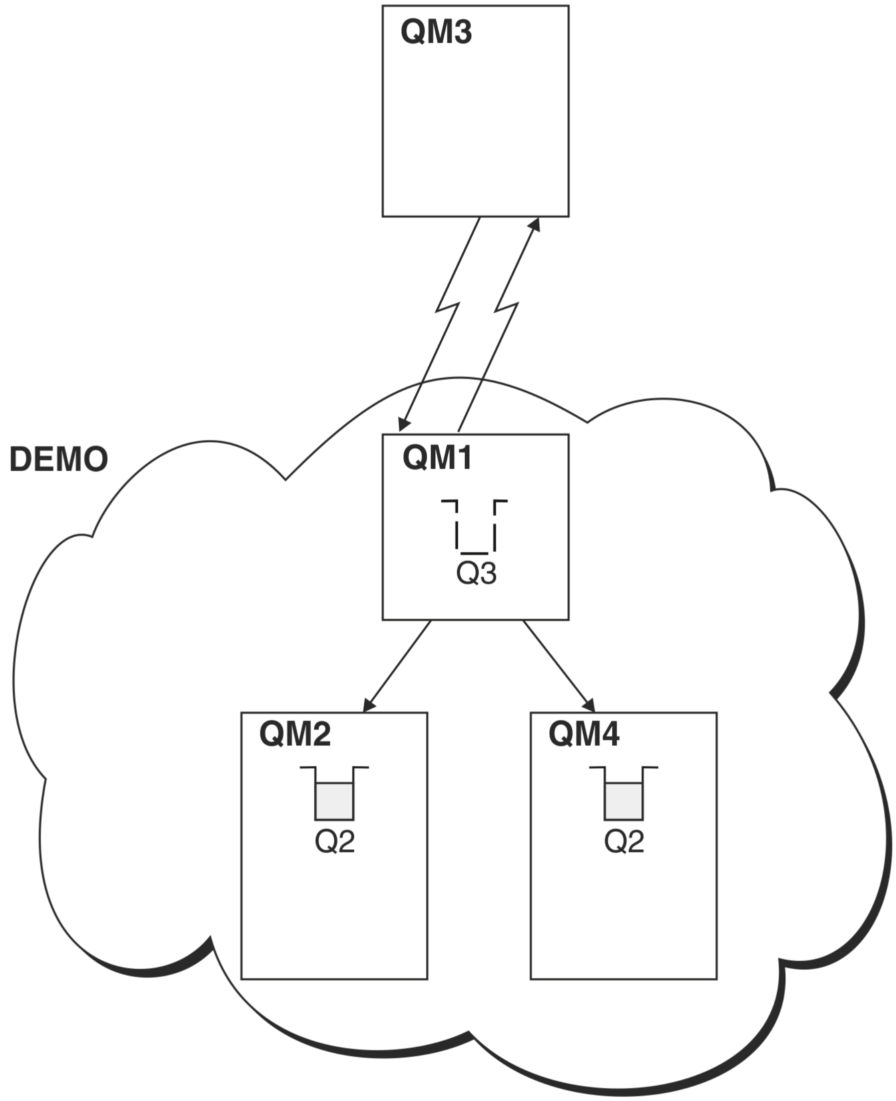

Figure 57. Putting from a queue manager outside the cluster

When an application at QM3 issues an MQPUT call to put a message to Q2, the QREMOTE definition on QM3 causes the message to be routed through the gateway queue manager QM1. When QM1 receives the message, it is aware that the message is still intended for a queue named Q2 and performs name resolution. QM1 checks its local definitions and does not find any for Q2. QM1 then checks its cluster configuration and finds that it is aware of two instances of Q2 in cluster DEMO. QM1 can now make use of workload balancing to distribute messages between the instances of Q2 residing on QM2 and QM4.

Related concepts

Queue-manager aliases and clusters
Use queue-manager aliases to hide the name of queue managers when sending messages into or out of a cluster, and to workload balance messages sent to a cluster.

Reply-to queue aliases and clusters
A reply-to queue alias definition is used to specify alternative names for reply information. Reply-to queue alias definitions can be used with clusters just the same as in a distributed queuing environment.

Queue aliases and clusters

Use queue aliases to hide the name of a cluster queue, to cluster a queue, adopt different attributes, or adopt different access controls.

Name resolution

Related tasks

Configuring request/reply to a cluster
Configure a request/reply message path from a queue manager outside a cluster. Hide the inner details of the cluster by using a gateway queue manager as the communication path to and from the cluster.

Configuring request/reply from a cluster
Configure a request/reply message path from a cluster to a queue manager outside the cluster. Hide the details of how a queue manager inside the cluster communicates outside the cluster by using a gateway queue manager.

Configuring message paths between clusters
Connect clusters together using a gateway queue manager. Make queues or queue managers visible to all the clusters by defining cluster queue or cluster queue manager aliases on the gateway queue manager.

Related reference

Queue name resolution

Configuring message paths between clusters

Connect clusters together using a gateway queue manager. Make queues or queue managers visible to all the clusters by defining cluster queue or cluster queue manager aliases on the gateway queue manager.

About this task

Instead of grouping all your queue managers together in one large cluster, you can have many smaller clusters. Each cluster has one or more queue managers in acting as a bridge. The advantage of this is that you can restrict the visibility of queue and queue-manager names across the clusters. See Overlapping clusters. Use aliases to change the names of queues and queue managers to avoid name conflicts or to comply with local naming conventions.

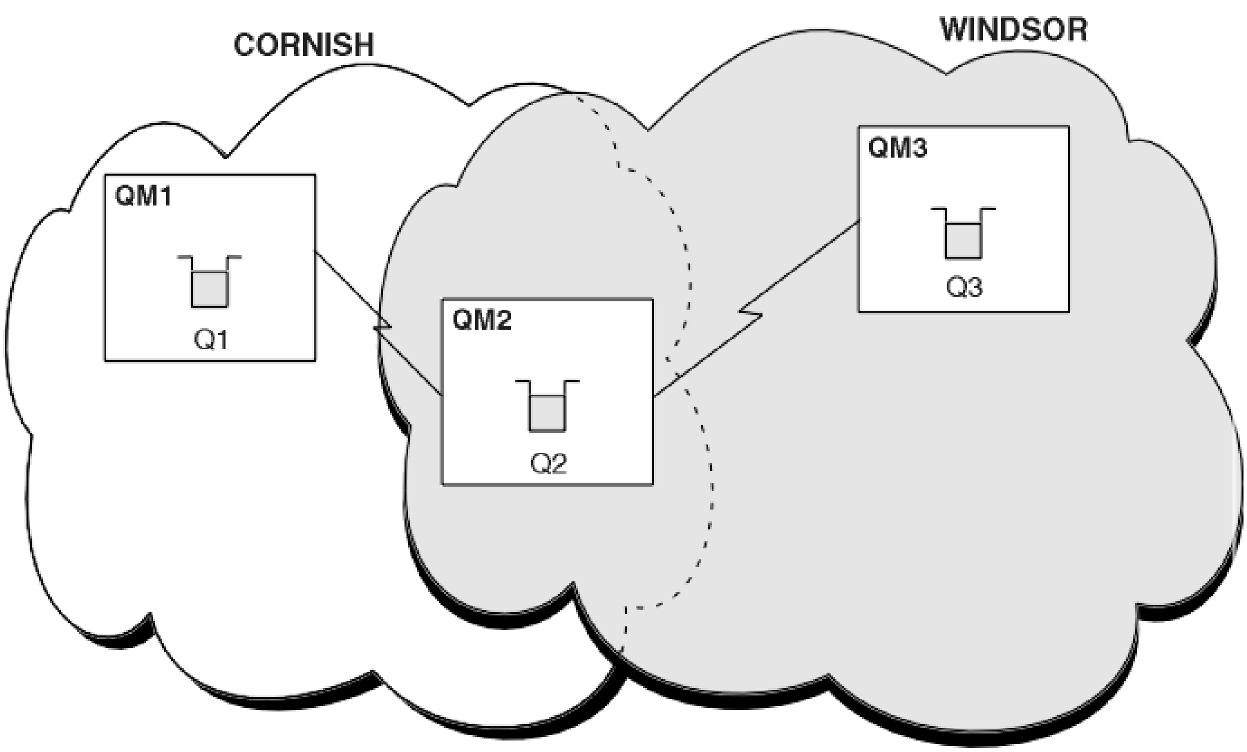

Figure 58. Bridging across clusters

Figure 58 on page 305 shows two clusters with a bridge between them. There could be more than one bridge.

Configure the clusters using the following procedure:

Procedure

1. Define a cluster queue, Q1 on QM1.

   ```
   DEFINE QLOCAL(Q1) CLUSTER(CORNISH)
   ```

2. Define a cluster queue, Q3 on QM3.

   ```
   DEFINE QLOCAL(Q3) CLUSTER(WINDSOR)
   ```

3. Create a namelist called CORNISHWINDSOR on QM2, containing the names of both clusters.

   ```
   DEFINE NAMELIST(CORNISHWINDSOR) DESCR('CornishWindsor namelist')
   NAMES(CORNISH, WINDSOR)
   ```

4. Define a cluster queue, Q2 on QM2

   ```
   DEFINE QLOCAL(Q2) CLUSNL(CORNISHWINDSOR)
   ```

What to do next

QM2 is a member of both clusters and is the bridge between them. For each queue that you want to make visible across the bridge, you need a QALIAS definition on the bridge. For example in Figure 58 on page 305, on QM2, you need:

```
DEFINE QALIAS(MYQ3) TARGET(Q3) CLUSTER(CORNISH) DEFBIND(NOTFIXED)
```

Using the queue alias, an application connected to a queue manager in CORNISH, for example QM1, can put a message to Q3. It refers to Q3 as MYQ3. The message is routed to Q3 at QM3.

When you open a queue, you need to set DEFBIND to either NOTFIXED or QDEF. If DEFBIND is left as the default, OPEN, the queue manager resolves the alias definition to the bridge queue manager that hosts it. The bridge does not forward the message.

For each queue manager that you want to make visible, you need a queue-manager alias definition. For example, on QM2 you need:

```
DEFINE QREMOTE(QM1) RNAME(' ') RQMNAME(QM1) CLUSTER(WINDSOR)
```

An application connected to any queue manager in WINDSOR, for example QM3, can put a message to any queue on QM1, by naming QM1 explicitly on the MQOPEN call.

Related concepts

Queue-manager aliases and clusters
Use queue-manager aliases to hide the name of queue managers when sending messages into or out of a cluster, and to workload balance messages sent to a cluster.

Reply-to queue aliases and clusters
A reply-to queue alias definition is used to specify alternative names for reply information. Reply-to queue alias definitions can be used with clusters just the same as in a distributed queuing environment.

Queue aliases and clusters

Use queue aliases to hide the name of a cluster queue, to cluster a queue, adopt different attributes, or adopt different access controls.

Related tasks

Configuring request/reply to a cluster
Configure a request/reply message path from a queue manager outside a cluster. Hide the inner details of the cluster by using a gateway queue manager as the communication path to and from the cluster.

Configuring request/reply from a cluster
Configure a request/reply message path from a cluster to a queue manager outside the cluster. Hide the details of how a queue manager inside the cluster communicates outside the cluster by using a gateway queue manager.

Configuring workload balancing from outside a cluster
Configure a message path from a queue manager outside a cluster to any copy of a cluster queue. The result is to workload balance requests from outside the cluster to each instance of a cluster queue.

Queue-manager aliases and clusters

Use queue-manager aliases to hide the name of queue managers when sending messages into or out of a cluster, and to workload balance messages sent to a cluster.

Queue-manager aliases, which are created using a remote-queue definition with a blank RNAME, have five uses:

Remapping the queue-manager name when sending messages

A queue-manager alias can be used to remap the queue-manager name specified in an MQOPEN call to another queue manager. It can be a cluster queue manager. For example, a queue manager might have the queue-manager alias definition:

```
DEFINE QREMOTE(YORK) RNAME(' ') RQMNAME(CLUSQM)
```

YORK can be used as an alias for the queue manager called CLUSQM. When an application on the queue manager that made this definition puts a message to queue manager YORK, the local queue manager resolves the name to CLUSQM. If the local queue manager is not called CLUSQM, it puts the message on the cluster transmission queue to be moved to CLUSQM. It also changes the transmission header to say CLUSQM instead of YORK.

Note: The definition applies only on the queue manager that makes it. To advertise the alias to the whole cluster, you need to add the CLUSTER attribute to the remote-queue definition. Then messages from other queue managers that were destined for YORK are sent to CLUSQM.

Altering or specifying the transmission queue when sending messages

Aliasing can be used to join a cluster to a non-cluster system. For example, queue managers in the cluster ITALY could communicate with the queue manager called PALERMO, which is outside the cluster. To communicate, one of the queue managers in the cluster must act as a gateway. From the gateway queue manager, issue the command:

```
DEFINE QREMOTE(ROME) RNAME(' ') RQMNAME(PALERMO) XMITQ(X) CLUSTER(ITALY)
```

The command is a queue-manager alias definition. It defines and advertises ROME as a queue manager over which messages from any queue manager in the cluster ITALY can multi-hop to reach their destination at PALERMO. Messages put to a queue opened with the queue-manager name set to ROME are sent to the gateway queue manager with the queue manager alias definition. Once there, the messages are put on the transmission queue X and moved by non-cluster channels to the queue manager PALERMO.

The choice of the name ROME in this example is not significant. The values for QREMOTE and RQMNAME could both be the same.

Determining the destination when receiving messages

When a queue manager receives a message, it extracts the name of the destination queue and queue manager from the transmission header. It looks for a queue-manager alias definition with the same name as the queue manager in the transmission header. If it finds one, it substitutes the RQMNAME from the queue-manager alias definition for the queue manager name in the transmission header.

There are two reasons for using a queue-manager alias in this way:

- To direct messages to another queue manager
- To alter the queue manager name to be the same as the local queue manager

Using queue manager aliases in a gateway queue manager to route messages between queue managers in different clusters.

An application can send a message to a queue in a different cluster using a queue manager alias. The queue does not have to be a cluster queue. The queue is defined in one cluster. The application is connected to a queue manager in a different cluster. A gateway queue manager connects the two clusters. If the queue is not defined as clustered, for the correct routing to take place, the application must open the queue using the queue name and a clustered queue manager alias name. For an example of a configuration, see "Creating two-overlapping clusters with a gateway queue manager" on page 263, from which the reply message flow illustrated in figure 1, is taken.

The diagram shows the path taken by the reply message back to a temporary dynamic queue, which is called RQ. The server application, connected to QM3, opens the reply queue using the queue manager name QM2. The queue manager name QM2 is defined as a clustered queue manager alias on QM1. QM3 routes the reply message to QM1. QM1 routes the message to QM2.

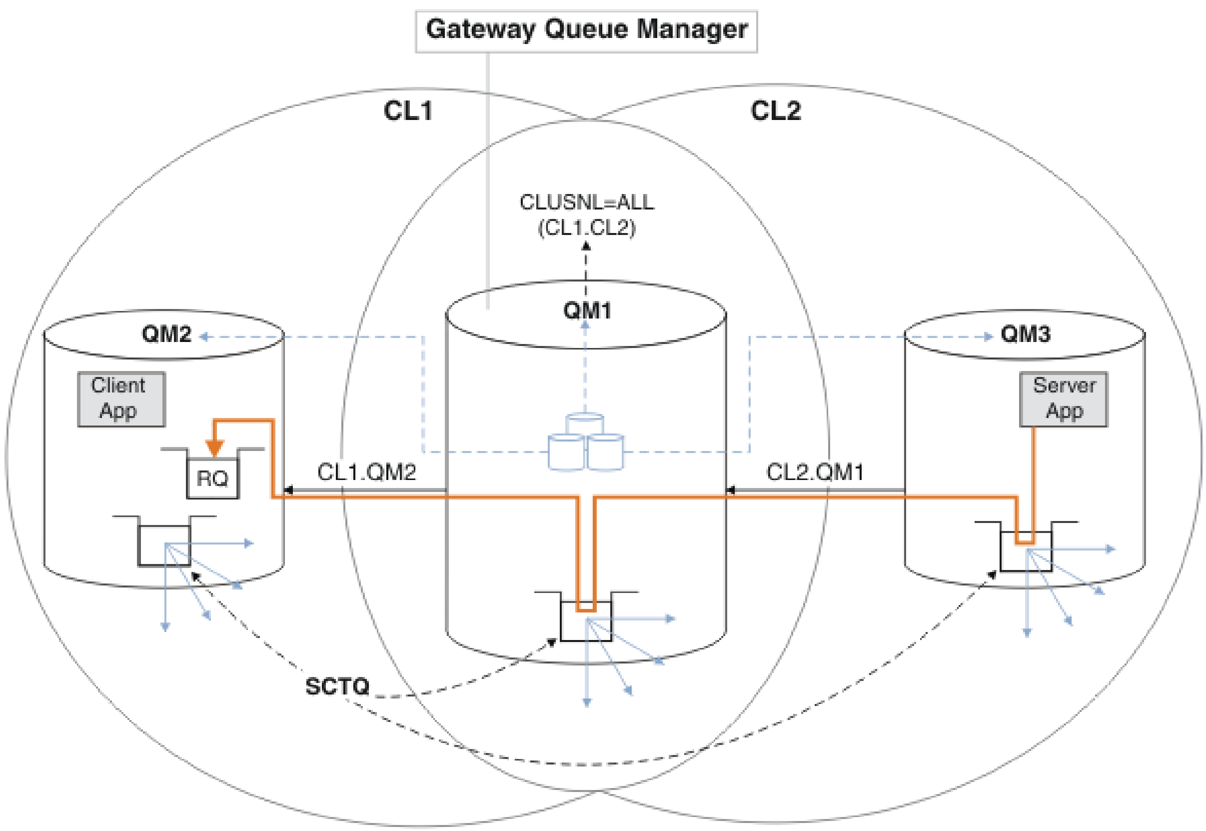

Figure 59. Using a queue manager alias to return the reply message to a different cluster

The way the routing works is as follows. Every queue manager in each cluster has a queue manager alias definition on QM1. The aliases are clustered in all the clusters. The grey dashed arrows from each of the aliases to a queue manager show that each queue manager alias is resolved to a real queue manager in at least one of the clusters. In this case, the QM2 alias is clustered in both cluster CL1 and

CL2, and is resolved to the real queue manager QM2 in CL1. The server application creates the reply message using the reply to queue name RQ, and reply to queue manager name QM2. The message is routed to QM1 because the queue manager alias definition QM2 is defined on QM1 in cluster CL2 and queue manager QM2 is not in cluster CL2. As the message cannot be sent to the target queue manager, it is sent to the queue manager that has the alias definition.

QM1 places the message on the cluster transmission queue on QM1 for transferal to QM2. QM1 routes the message to QM2 because the queue manager alias definition on QM1 for QM2 defines QM2 as the real target queue manager. The definition is not circular, because alias definitions can refer only to real definitions; the alias cannot point to itself. The real definition is resolved by QM1, because both QM1 and QM2 are in the same cluster, CL1. QM1 finds out the connection information for QM2 from the repository for CL1, and routes the message to QM2. For the message to be rerouted by QM1, the server application must have opened the reply queue with the option DEFBIND set to MQBND_BIND_NOT_FIXED. If the server application had opened the reply queue with the option MQBND_BIND_ON_OPEN, the message is not rerouted and ends up on a dead letter queue.

Using a queue manager as a gateway into the cluster to workload balance messages from coming from outside the cluster.

You define a queue called EDINBURGH on more than one queue manager in the cluster. You want the clustering mechanism to balance the workload for messages coming to that queue from outside the cluster.

A queue manager from outside the cluster needs a transmit queue and sender-channel to one queue manager in the cluster. This queue is called a gateway queue manager. To take advantage of the default workload balancing mechanism, one of the following rules must apply:

- The gateway queue manager must not contain an instance of the EDINBURGH queue.
- The gateway queue manager specifies CLWLUSEQ(ANY) on ALTER QMGR.

For an example of workload balancing from outside a cluster, see "Configuring workload balancing from outside a cluster" on page 303

Related concepts

Reply-to queue aliases and clusters
A reply-to queue alias definition is used to specify alternative names for reply information. Reply-to queue alias definitions can be used with clusters just the same as in a distributed queuing environment.

Queue aliases and clusters
Use queue aliases to hide the name of a cluster queue, to cluster a queue, adopt different attributes, or adopt different access controls.

Related tasks

Configuring request/reply to a cluster
Configure a request/reply message path from a queue manager outside a cluster. Hide the inner details of the cluster by using a gateway queue manager as the communication path to and from the cluster.

Configuring request/reply from a cluster
Configure a request/reply message path from a cluster to a queue manager outside the cluster. Hide the details of how a queue manager inside the cluster communicates outside the cluster by using a gateway queue manager.

Configuring workload balancing from outside a cluster
Configure a message path from a queue manager outside a cluster to any copy of a cluster queue. The result is to workload balance requests from outside the cluster to each instance of a cluster queue.

Configuring message paths between clusters

Connect clusters together using a gateway queue manager. Make queues or queue managers visible to all the clusters by defining cluster queue or cluster queue manager aliases on the gateway queue manager.

Reply-to queue aliases and clusters

A reply-to queue alias definition is used to specify alternative names for reply information. Reply-to queue alias definitions can be used with clusters just the same as in a distributed queuing environment.

For example:

- An application at queue manager VENICE sends a message to queue manager PISA using the MQPUT call. The application provides the following reply-to queue information in the message descriptor:

```
ReplyToQ='QUEUE'
ReplyToQMgr=''
```

- In order that replies sent to QUEUE can be received on OTHERQ at PISA, create a remote-queue definition on VENICE that is used as a reply-to queue alias. The alias is effective only on the system on which it was created.

```
DEFINE QREMOTE(QUEUE) RNAME(OTHERQ) RQMNAME(PISA)
```

RQMNAME and QREMOTE can specify the same names, even if RQMNAME is itself a cluster queue manager.

Related concepts

Queue-manager aliases and clusters
Use queue-manager aliases to hide the name of queue managers when sending messages into or out of a cluster, and to workload balance messages sent to a cluster.

Queue aliases and clusters
Use queue aliases to hide the name of a cluster queue, to cluster a queue, adopt different attributes, or adopt different access controls.

Related tasks

Configuring request/reply to a cluster
Configure a request/reply message path from a queue manager outside a cluster. Hide the inner details of the cluster by using a gateway queue manager as the communication path to and from the cluster.

Configuring request/reply from a cluster
Configure a request/reply message path from a cluster to a queue manager outside the cluster. Hide the details of how a queue manager inside the cluster communicates outside the cluster by using a gateway queue manager.

Configuring workload balancing from outside a cluster
Configure a message path from a queue manager outside a cluster to any copy of a cluster queue. The result is to workload balance requests from outside the cluster to each instance of a cluster queue.

Configuring message paths between clusters
Connect clusters together using a gateway queue manager. Make queues or queue managers visible to all the clusters by defining cluster queue or cluster queue manager aliases on the gateway queue manager.

Queue aliases and clusters

Use queue aliases to hide the name of a cluster queue, to cluster a queue, adopt different attributes, or adopt different access controls.

A QALIAS definition is used to create an alias by which a queue is to be known. You might create an alias for a number of reasons:

- You want to start using a different queue but you do not want to change your applications.
- You do not want applications to know the real name of the queue to which they are putting messages.
- You might have a naming convention that differs from the one where the queue is defined.
- Your applications might not be authorized to access the queue by its real name but only by its alias.

Create a QALIAS definition on a queue manager using the DEFINE QALIAS command. For example, run the command:

```
DEFINE QALIAS(PUBLIC) TARGET(LOCAL) CLUSTER(C)
```

The command advertises a queue called PUBLIC to the queue managers in cluster C. PUBLIC is an alias that resolves to the queue called LOCAL. Messages sent to PUBLIC are routed to the queue called LOCAL.

You can also use a queue alias definition to resolve a queue name to a cluster queue. For example, run the command:

```
DEFINE QALIAS(PRIVATE) TARGET(PUBLIC)
```

The command enables a queue manager to use the name PRIVATE to access a queue advertised elsewhere in the cluster by the name PUBLIC. Because this definition does not include the CLUSTER attribute it applies only to the queue manager that makes it.

Related concepts

Queue-manager aliases and clusters
Use queue-manager aliases to hide the name of queue managers when sending messages into or out of a cluster, and to workload balance messages sent to a cluster.

Reply-to queue aliases and clusters
A reply-to queue alias definition is used to specify alternative names for reply information. Reply-to queue alias definitions can be used with clusters just the same as in a distributed queuing environment.

Related tasks

Configuring request/reply to a cluster
Configure a request/reply message path from a queue manager outside a cluster. Hide the inner details of the cluster by using a gateway queue manager as the communication path to and from the cluster.

Configuring request/reply from a cluster
Configure a request/reply message path from a cluster to a queue manager outside the cluster. Hide the details of how a queue manager inside the cluster communicates outside the cluster by using a gateway queue manager.

Configuring workload balancing from outside a cluster
Configure a message path from a queue manager outside a cluster to any copy of a cluster queue. The result is to workload balance requests from outside the cluster to each instance of a cluster queue.

Configuring message paths between clusters
Connect clusters together using a gateway queue manager. Make queues or queue managers visible to all the clusters by defining cluster queue or cluster queue manager aliases on the gateway queue manager.

Using clusters for workload management

By defining multiple instances of a queue on different queue managers in a cluster you can spread the work of servicing the queue over multiple servers. There are several factors that can prevent messages being requeued to a different queue manager in the event of failure.

As well as setting up clusters to reduce system administration, you can create clusters in which more than one queue manager hosts an instance of the same queue.

You can organize your cluster such that the queue managers in it are clones of each other. Each queue manager is able to run the same applications and have local definitions of the same queues. z/OS For example, in a z/OS parallel sysplex the cloned applications might access data in a shared Db2 or Virtual Storage Access Method (VSAM) database. You can spread the workload between your queue managers by having several instances of an application. Each instance of the application receives messages and runs independently of the others.

The advantages of using clusters in this way are as follows:

- Increased availability of your queues and applications.
- Faster throughput of messages.
- More even distribution of workload in your network.

Any one of the queue managers that hosts an instance of a particular queue can handle messages destined for that queue, and applications do not name a queue manager when sending messages. If a cluster contains more than one instance of the same queue, IBM MQ selects a queue manager to route a message to. Suitable destinations are chosen based on the availability of the queue manager and queue, and on a number of cluster workload-specific attributes associated with queue managers, queues, and channels. See Workload balancing in clusters.

z/OS In IBM MQ for z/OS, queue managers that are in queue-sharing groups can host cluster queues as shared queues. Shared cluster queues are available to all queue managers in the same queue-sharing group. For example, in A cluster with multiple instances of the same queue, either or both of the queue managers QM2 and QM4 can be a shared-queue manager. Each has a definition for the queue Q3. Any of the queue managers in the same queue-sharing group as QM4 can read a message put to the shared queue Q3. Each queue-sharing group can contain up to 32 queue managers, each with access to the same data. Queue sharing significantly increases the throughput of your messages.

See the following subtopics for more information about cluster configurations for workload management:

Related concepts

Comparison of clustering and distributed queuing

Distributed queuing and clusters

Components of a cluster

Cluster channels

What happens if a cluster queue is disabled for MQPUT

Workload balancing set on a cluster-sender channel is not working

“Routing messages to and from clusters” on page 293
Use queue aliases, queue manager aliases, and remote queue definitions to connect clusters to external queue managers and other clusters.

Related tasks

Writing and compiling cluster workload exits

“Configuring a queue manager cluster” on page 215
Clusters provide a mechanism for interconnecting queue managers in a way that simplifies both the initial configuration and the ongoing management. You can define cluster components, and create and manage clusters.

“Setting up a new cluster” on page 227
Follow these instructions to set up the example cluster. Separate instructions describe setting up the cluster on TCP/IP, LU 6.2, and with a single transmission queue or multiple transmission queues. Test the cluster works by sending a message from one queue manager to the other.

“Configuring workload balancing from outside a cluster” on page 303
Configure a message path from a queue manager outside a cluster to any copy of a cluster queue. The result is to workload balance requests from outside the cluster to each instance of a cluster queue.

Related reference

The Cluster Queue Monitoring sample program (AMQSCLM)

Example of a cluster with more than one instance of a queue

In this example of a cluster with more than one instance of a queue, messages are routed to different instances of the queue. You can force a message to a specific instance of the queue, and you can choose to send a sequence of messages to one of either of the queue managers.

Figure 60 on page 313 shows a cluster in which there is more than one definition for the queue Q3. If an application at QM1 puts a message to Q3, it does not necessarily know which instance of Q3 is going to process its message. If an application is running on QM2 or QM4, where there are local instances of Q3, the local instance of Q3 is opened by default. By setting the CLWLUSEQ queue attribute, the local instance of the queue can be treated the same as a remote instance of the queue.

The MQOPEN option DefBind controls whether the target queue manager is chosen when the MQOPEN call is issued, or when the message is transferred from the transmission queue.

If you set DefBind to MQBND_BIND_NOT_FIXED the message can be sent to an instance of the queue that is available when the message is transmitted. This avoids the following problems:

- The target queue is unavailable when the message arrives at the target queue manager.
- The state of the queue has changed.
- The message has been put using a cluster queue alias, and no instance of the target queue exists on the queue manager where the instance of the cluster queue alias is defined.

If any if these problems are discovered at transmission time, another available instance of the target queue is sought and the message is rerouted. If no instances of the queue are available, the message is placed on the dead-letter queue.

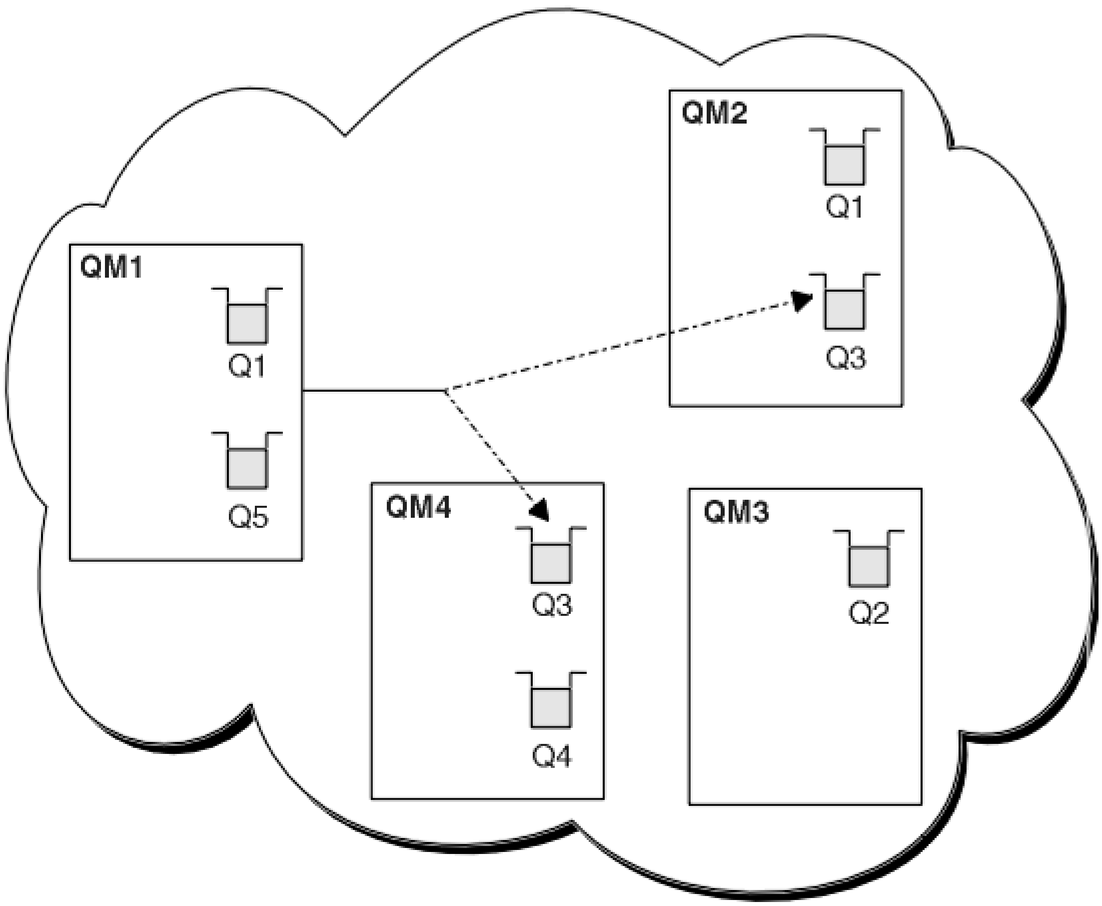

Figure 60. A cluster with multiple instances of the same queue

One factor that can prevent messages being rerouted is if messages have been assigned to a fixed queue manager or channel with MQBND_BIND_ON_OPEN. Messages bound on MQOPEN are never reallocated to another channel. Note also that message reallocation only takes place when a cluster channel is actually failing. Reallocation does not occur if the channel has already failed.

The system attempts to reroute a message if the destination queue manager goes out of service. In so doing, it does not affect the integrity of the message by running the risk of losing it or by creating a duplicate. If a queue manager fails and leaves a message in doubt, that message is not rerouted.

z/OS On IBM MQ for z/OS, the channel does not completely stop until the message reallocation process is complete. Stopping the channel with mode set to FORCE or TERMINATE does interrupt the process, so if you do this then some BIND_NOT_FIXED messages might have already been reallocated to another channel, or the messages might be out of order.

Note: z/OS

1. Before setting up a cluster that has multiple instances of the same queue, ensure that your messages do not have dependencies on each other. For example, needing to be processed in a specific sequence or by the same queue manager.
2. Make the definitions for different instances of the same queue identical. Otherwise you get different results from different MQINQ calls.

Related concepts

Application programming and clusters
You do not need to make any programming changes to take advantage of multiple instances of the same queue. However, some programs do not work correctly unless a sequence of messages is sent to the same instance of a queue.

Related tasks

Adding a queue manager that hosts a queue locally
Follow these instructions to add an instance of INVENTQ to provide additional capacity to run the inventory application system in Paris and New York.

Using two networks in a cluster
Follow these instructions to add a new store in TOKYO where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

Using a primary and a secondary network in a cluster
Follow these instructions to make one network the primary network, and another network the backup network. Use the backup network if there is a problem with the primary network.

Adding a queue to act as a backup
Follow these instructions to provide a backup in Chicago for the inventory system that now runs in New York. The Chicago system is only used when there is a problem with the New York system.

Restricting the number of channels used
Follow these instructions to restrict the number of active channels each server runs when a price check application is installed on various queue managers.

Adding a more powerful queue manager that hosts a queue
Follow these instructions to provide additional capacity by running the inventory system in Los Angeles as well as New York, where Los Angeles can handle twice the number of messages as New York.

Adding a queue manager that hosts a queue locally

Follow these instructions to add an instance of INVENTQ to provide additional capacity to run the inventory application system in Paris and New York.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The INVENTORY cluster has been set up as described in Adding a new queue manager to a cluster. It contains three queue managers; LONDON and NEWYORK both hold full repositories, PARIS holds a partial repository. The inventory application runs on the system in New York, connected to the NEWYORK queue manager. The application is driven by the arrival of messages on the INVENTQ queue.
- We want to add an instance of INVENTQ to provide additional capacity to run the inventory application system in Paris and New York.

About this task

Follow these steps to add a queue manager that hosts a queue locally.

Procedure

1. Alter the PARIS queue manager.

   For the application in Paris to use the INVENTQ in Paris and the one in New York, we must inform the queue manager. On PARIS issue the following command:

   ```
   ALTER QMGR CLWLUSEQ(ANY)
   ```
2. Review the inventory application for message affinities.

   Before proceeding, ensure that the inventory application does not have any dependencies on the sequence of processing of messages. For more information, see Handling message affinities.
3. Install the inventory application on the system in Paris.
4. Define the cluster queue INVENTQ.

   The INVENTQ queue which is already hosted by the NEWYORK queue manager is also to be hosted by PARIS. Define it on the PARIS queue manager as follows:

   ```
   DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
   ```

   Now that you have completed all the definitions, if you have not already done so, start the channel initiator on IBM MQ for z/OS. On all platforms, start a listener program on queue manager PARIS. The listener listens for incoming network requests and starts the cluster-receiver channel when it is needed.

Results

Figure 61 on page 315 shows the cluster set up by this task.

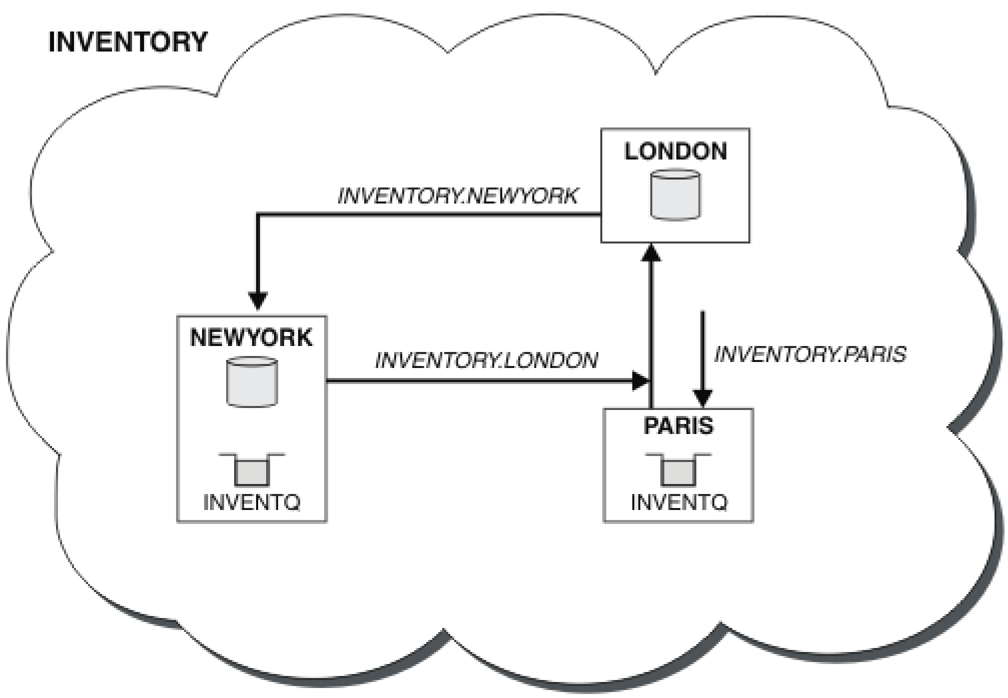

Figure 61. The INVENTORY cluster, with three queue managers

The modification to this cluster was accomplished without you altering the queue managers NEWYORK or LONDON. The full repositories in these queue managers are updated automatically with the information they need to be able to send messages to INVENTQ at PARIS.

What to do next

The INVENTQ queue and the inventory application are now hosted on two queue managers in the cluster. This increases their availability, speeds up throughput of messages, and allows the workload to be distributed between the two queue managers. Messages put to INVENTQ by any of the queue managers LONDON, NEWYORK, PARIS are routed alternately to PARIS or NEWYORK, so that the workload is balanced.

Related concepts

Example of a cluster with more than one instance of a queue
In this example of a cluster with more than one instance of a queue, messages are routed to different instances of the queue. You can force a message to a specific instance of the queue, and you can choose to send a sequence of messages to one of either of the queue managers.

Application programming and clusters
You do not need to make any programming changes to take advantage of multiple instances of the same queue. However, some programs do not work correctly unless a sequence of messages is sent to the same instance of a queue.

Related tasks

Using two networks in a cluster
Follow these instructions to add a new store in TOKYO where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

Using a primary and a secondary network in a cluster
Follow these instructions to make one network the primary network, and another network the backup network. Use the backup network if there is a problem with the primary network.

Adding a queue to act as a backup
Follow these instructions to provide a backup in Chicago for the inventory system that now runs in New York. The Chicago system is only used when there is a problem with the New York system.

Restricting the number of channels used
Follow these instructions to restrict the number of active channels each server runs when a price check application is installed on various queue managers.

Adding a more powerful queue manager that hosts a queue
Follow these instructions to provide additional capacity by running the inventory system in Los Angeles as well as New York, where Los Angeles can handle twice the number of messages as New York.

Using two networks in a cluster

Follow these instructions to add a new store in TOKYO where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The INVENTORY cluster has been set up as described in "Adding a queue manager to a cluster". It contains three queue managers; LONDON and NEWYORK both hold full repositories, PARIS holds a partial repository. The inventory application runs on the system in New York, connected to the NEWYORK queue manager. The application is driven by the arrival of messages on the INVENTQ queue.
- A new store is being added in TOKYO where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

About this task

Follow these steps to use two networks in a cluster.

Procedure

1. Decide which full repository TOKYO refers to first.

   Every queue manager in a cluster must refer to one or other of the full repositories to gather information about the cluster. It builds up its own partial repository. It is of no particular significance which repository you choose. In this example, NEWYORK is chosen. Once the new queue manager has joined the cluster it communicates with both of the repositories.

2. Define the CLUSRCVR channels.

   Every queue manager in a cluster needs to define a cluster-receiver on which it can receive messages. This queue manager needs to be able to communicate on each network.

   ```
   DEFINE CHANNEL(INVENTORY.TOKYO.NETB) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME('TOKYO.NETB.CMSTORE.COM') CLUSTER(INVENTORY) DESCR('Cluster-receiver
   channel using network B for TOKYO')
   ```

   ```
   DEFINE CHANNEL(INVENTORY.TOKYO.NETA) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME('TOKYO.NETA.CMSTORE.COM') CLUSTER(INVENTORY) DESCR('Cluster-receiver
   channel using network A for TOKYO')
   ```

3. Define a CLUSSDR channel on queue manager TOKYO.

   Every queue manager in a cluster needs to define one cluster-sender channel on which it can send messages to its first full repository. In this case we have chosen NEWYORK, so TOKYO needs the following definition:

   ```
   DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY) DESCR('Cluster-sender
   channel from TOKYO to repository at NEWYORK')
   ```

   Now that you have completed all the definitions, if you have not already done so start the channel initiator on IBM MQ for z/OS. On all platforms, start a listener program on queue manager PARIS. The listener program listens for incoming network requests and starts the cluster-receiver channel when it is needed.

Results

Figure 62 on page 318 shows the cluster set up by this task.

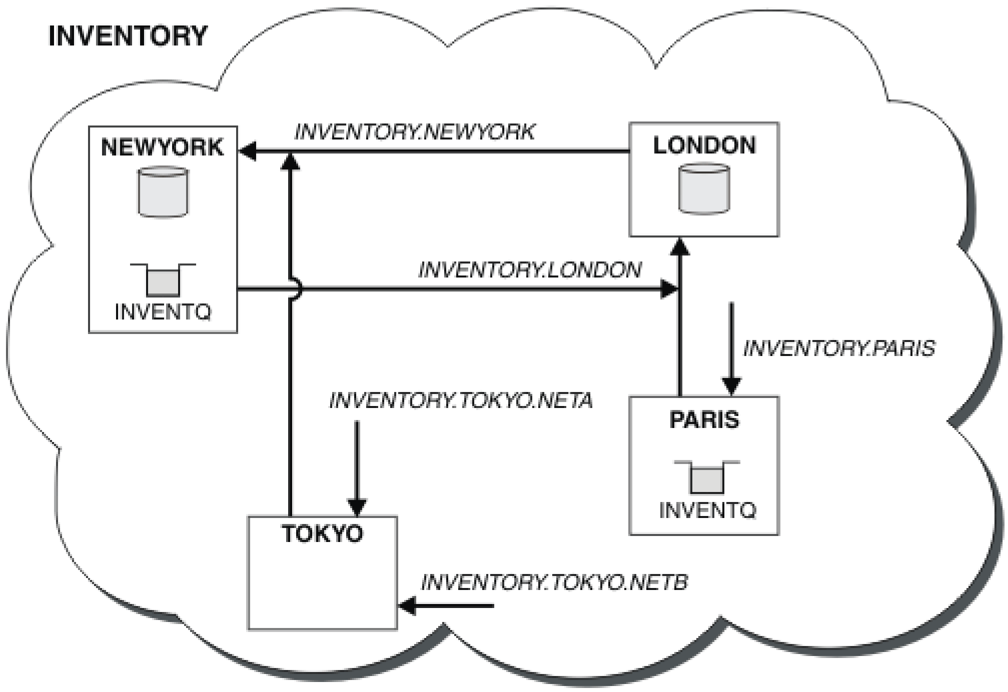

Figure 62. The `INVENTORY` cluster, with four queue managers

By making only three definitions, we have added the queue manager `TOKYO` to the cluster with two different network routes available to it.

Related concepts

Example of a cluster with more than one instance of a queue
In this example of a cluster with more than one instance of a queue, messages are routed to different instances of the queue. You can force a message to a specific instance of the queue, and you can choose to send a sequence of messages to one of either of the queue managers.

Application programming and clusters
You do not need to make any programming changes to take advantage of multiple instances of the same queue. However, some programs do not work correctly unless a sequence of messages is sent to the same instance of a queue.

Related tasks

Adding a queue manager that hosts a queue locally
Follow these instructions to add an instance of `INVENTQ` to provide additional capacity to run the inventory application system in Paris and New York.

Using a primary and a secondary network in a cluster
Follow these instructions to make one network the primary network, and another network the backup network. Use the backup network if there is a problem with the primary network.

Adding a queue to act as a backup
Follow these instructions to provide a backup in Chicago for the inventory system that now runs in New York. The Chicago system is only used when there is a problem with the New York system.

Restricting the number of channels used
Follow these instructions to restrict the number of active channels each server runs when a price check application is installed on various queue managers.

Adding a more powerful queue manager that hosts a queue
Follow these instructions to provide additional capacity by running the inventory system in Los Angeles as well as New York, where Los Angeles can handle twice the number of messages as New York.

“Adding a queue manager to a cluster” on page 237

Follow these instructions to add a queue manager to the cluster you created. Messages to cluster queues and topics are transferred using the single cluster transmission queue `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.

Using a primary and a secondary network in a cluster

Follow these instructions to make one network the primary network, and another network the backup network. Use the backup network if there is a problem with the primary network.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The `INVENTORY` cluster has been set up as described in "Using two networks in a cluster" on page 316. It contains four queue managers; `LONDON` and `NEWYORK` both hold full repositories; `PARIS` and `TOKYO` hold partial repositories. The inventory application runs on the system in New York, connected to the queue manager `NEWYORK`. The `TOKYO` queue manager has two different networks that it can communicate on.
- You want to make one of the networks the primary network, and another of the networks the backup network. You plan to use the backup network if there is a problem with the primary network.

About this task

Use the `NETPRTY` attribute to configure a primary and a secondary network in a cluster.

Procedure

Alter the existing `CLUSRCVR` channels on `TOKYO`.

To indicate that the network A channel is the primary channel, and the network B channel is the secondary channel, use the following commands:

a) `ALTER CHANNEL(INVENTORY.TOKYO.NETA) CHLTYPE(CLUSRCVR) NETPRTY(2) DESCR('Main cluster-receiver channel for TOKYO')`

b) `ALTER CHANNEL(INVENTORY.TOKYO.NETB) CHLTYPE(CLUSRCVR) NETPRTY(1) DESCR('Backup cluster-receiver channel for TOKYO')`

What to do next

By configuring the channel with different network priorities, you have now defined to the cluster that you have a primary network and a secondary network. The queue managers in the cluster that use these channels automatically use the primary network whenever it is available. The queue managers failover to use the secondary network when the primary network is not available.

Related concepts

Example of a cluster with more than one instance of a queue
In this example of a cluster with more than one instance of a queue, messages are routed to different instances of the queue. You can force a message to a specific instance of the queue, and you can choose to send a sequence of messages to one of either of the queue managers.

Application programming and clusters
You do not need to make any programming changes to take advantage of multiple instances of the same queue. However, some programs do not work correctly unless a sequence of messages is sent to the same instance of a queue.

Related tasks

Adding a queue manager that hosts a queue locally

Follow these instructions to add an instance of INVENTQ to provide additional capacity to run the inventory application system in Paris and New York.

Using two networks in a cluster
Follow these instructions to add a new store in TOKYO where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

Adding a queue to act as a backup
Follow these instructions to provide a backup in Chicago for the inventory system that now runs in New York. The Chicago system is only used when there is a problem with the New York system.

Restricting the number of channels used
Follow these instructions to restrict the number of active channels each server runs when a price check application is installed on various queue managers.

Adding a more powerful queue manager that hosts a queue
Follow these instructions to provide additional capacity by running the inventory system in Los Angeles as well as New York, where Los Angeles can handle twice the number of messages as New York.

Adding a queue to act as a backup

Follow these instructions to provide a backup in Chicago for the inventory system that now runs in New York. The Chicago system is only used when there is a problem with the New York system.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The INVENTORY cluster has been set up as described in "Adding a queue manager to a cluster" on page 237. It contains three queue managers; LONDON and NEWYORK both hold full repositories, PARIS holds a partial repository. The inventory application runs on the system in New York, connected to the NEWYORK queue manager. The application is driven by the arrival of messages on the INVENTQ queue.
- A new store is being set up in Chicago to provide a backup for the inventory system that now runs in New York. The Chicago system only used when there is a problem with the New York system.

About this task

Follow these steps to add a queue to act as a backup.

Procedure

1. Decide which full repository CHICAGO refers to first.

   Every queue manager in a cluster must refer to one or other of the full repositories to gather information about the cluster. It builds up its own partial repository. It is of no particular significance which repository you choose for any particular queue manager. In this example, NEWYORK is chosen. Once the new queue manager has joined the cluster it communicates with both of the repositories.
2. Define the CLUSRCVR channel.

   Every queue manager in a cluster needs to define a cluster-receiver on which it can receive messages. On CHICAGO, define:

   ```
   DEFINE CHANNEL(INVENTORY.CHICAGO) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(CHICAGO.CMSTORE.COM) CLUSTER(INVENTORY) DESCR('Cluster-receiver
   channel for CHICAGO')
   ```
3. Define a CLUSSDR channel on queue manager CHICAGO.

   Every queue manager in a cluster needs to define one cluster-sender channel on which it can send messages to its first full repository. In this case we have chosen NEWYORK, so CHICAGO needs the following definition:

```
DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY) DESCR('Cluster-sender
channel from CHICAGO to repository at NEWYORK')
```

4. Alter the existing cluster queue INVENTQ.

   The INVENTQ which is already hosted by the NEWYORK queue manager is the main instance of the queue.

   ```
   ALTER QLOCAL(INVENTQ) CLWLPRTY(2)
   ```

5. Review the inventory application for message affinities.

   Before proceeding, ensure that the inventory application does not have any dependencies on the sequence of processing of messages.

6. Install the inventory application on the system in CHICAGO.
7. Define the backup cluster queue INVENTQ

   The INVENTQ which is already hosted by the NEWYORK queue manager, is also to be hosted as a backup by CHICAGO. Define it on the CHICAGO queue manager as follows:

   ```
   DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY) CLWLPRTY(1)
   ```

   Now that you have completed all the definitions, if you have not already done so start the channel initiator on IBM MQ for z/OS. On all platforms, start a listener program on queue manager CHICAGO. The listener program listens for incoming network requests and starts the cluster-receiver channel when it is needed.

Results

Figure 63 on page 321 shows the cluster set up by this task.

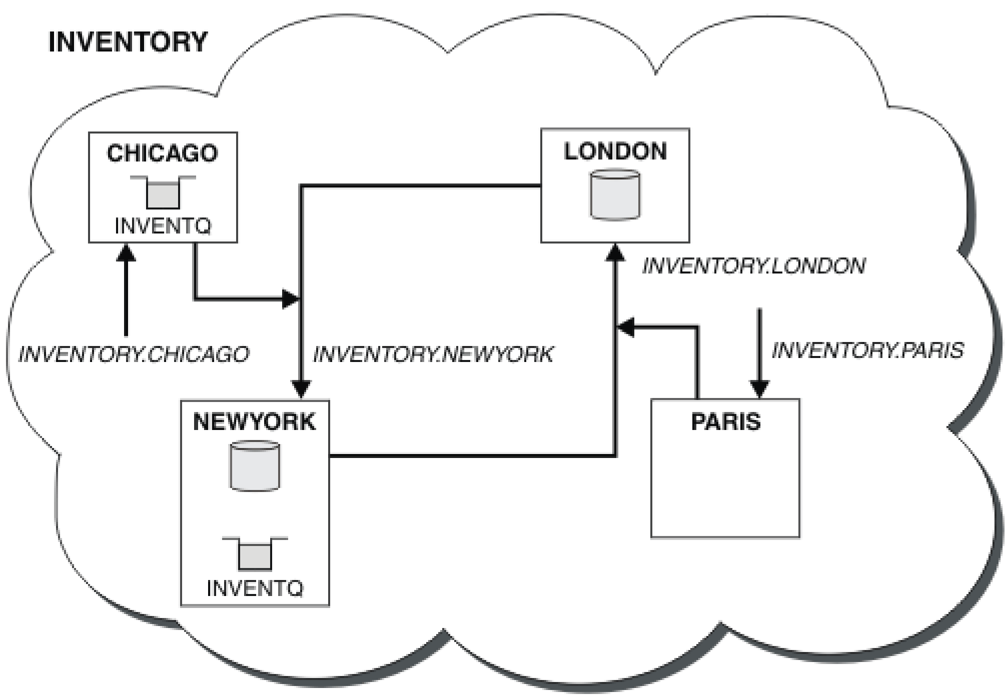

Figure 63. The INVENTORY cluster, with four queue managers

The INVENTQ queue and the inventory application are now hosted on two queue managers in the cluster. The CHICAGO queue manager is a backup. Messages put to INVENTQ are routed to NEWYORK unless it is unavailable when they are sent instead to CHICAGO.

Note:

The availability of a remote queue manager is based on the status of the channel to that queue manager. When channels start, their state changes several times, with some of the states being less preferential to the cluster workload management algorithm. In practice this means that lower priority (backup) destinations can be chosen while the channels to higher priority (primary) destinations are starting.

If you need to ensure that no messages go to a backup destination, do not use `CLWLPRTY`. Consider using separate queues, or `CLWLRANK` with a manual switch over from the primary to backup.

Related concepts

Example of a cluster with more than one instance of a queue
In this example of a cluster with more than one instance of a queue, messages are routed to different instances of the queue. You can force a message to a specific instance of the queue, and you can choose to send a sequence of messages to one of either of the queue managers.

Application programming and clusters
You do not need to make any programming changes to take advantage of multiple instances of the same queue. However, some programs do not work correctly unless a sequence of messages is sent to the same instance of a queue.

Related tasks

Adding a queue manager that hosts a queue locally
Follow these instructions to add an instance of `INVENTQ` to provide additional capacity to run the inventory application system in Paris and New York.

Using two networks in a cluster
Follow these instructions to add a new store in `TOKYO` where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

Using a primary and a secondary network in a cluster
Follow these instructions to make one network the primary network, and another network the backup network. Use the backup network if there is a problem with the primary network.

Restricting the number of channels used
Follow these instructions to restrict the number of active channels each server runs when a price check application is installed on various queue managers.

Adding a more powerful queue manager that hosts a queue
Follow these instructions to provide additional capacity by running the inventory system in Los Angeles as well as New York, where Los Angeles can handle twice the number of messages as New York.

Restricting the number of channels used

Follow these instructions to restrict the number of active channels each server runs when a price check application is installed on various queue managers.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- A price check application is to be installed on various queue managers. To keep the number of channels being used to a low number, the number of active channels each server runs is restricted. The application is driven by the arrival of messages on the `PRICEQ` queue.
- Four server queue managers host the price check application. Two query queue managers send messages to the `PRICEQ` to query a price. Two more queue managers are configured as full repositories.

About this task

Follow these steps to restrict the number of channels used.

Procedure

1. Choose two full repositories.

   Choose two queue managers to be the full repositories for your price check cluster. They are called REPOS1 and REPOS2.

   Issue the following command:

   ```
   ALTER QMGR REPOS(PRICECHECK)
   ```

2. Define a CLUSRCVR channel on each queue manager.

   At each queue manager in the cluster, define a cluster-receiver channel and a cluster-sender channel. It does not matter which is defined first.

   ```
   DEFINE CHANNEL(PRICECHECK.SERVE1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
   CONNAME(SERVER1.COM) CLUSTER(PRICECHECK) DESCR('Cluster-receiver channel')
   ```

3. Define a CLUSSDR channel on each queue manager.

   Make a CLUSSDR definition at each queue manager to link that queue manager to one or other of the full repository queue managers.

   ```
   DEFINE CHANNEL(PRICECHECK.REPOS1) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
   CONNAME(REPOS1.COM) CLUSTER(PRICECHECK) DESCR('Cluster-sender channel to
   repository queue manager')
   ```

4. Install the price check application.
5. Define the PRICEQ queue on all the server queue managers.

   Issue the following command on each:

   ```
   DEFINE QLOCAL(PRICEQ) CLUSTER(PRICECHECK)
   ```

6. Restrict the number of channels used by queries

   On the query queue managers we restrict the number of active channels used, by issuing the following commands on each:

   ```
   ALTER QMGR CLWLMRUC(2)
   ```

7. If you have not already done so, start the channel initiator on IBM MQ for z/OS. On all platforms, start a listener program.

   The listener program listens for incoming network requests and starts the cluster-receiver channel when it is needed.

Results

Figure 64 on page 324 shows the cluster set up by this task.

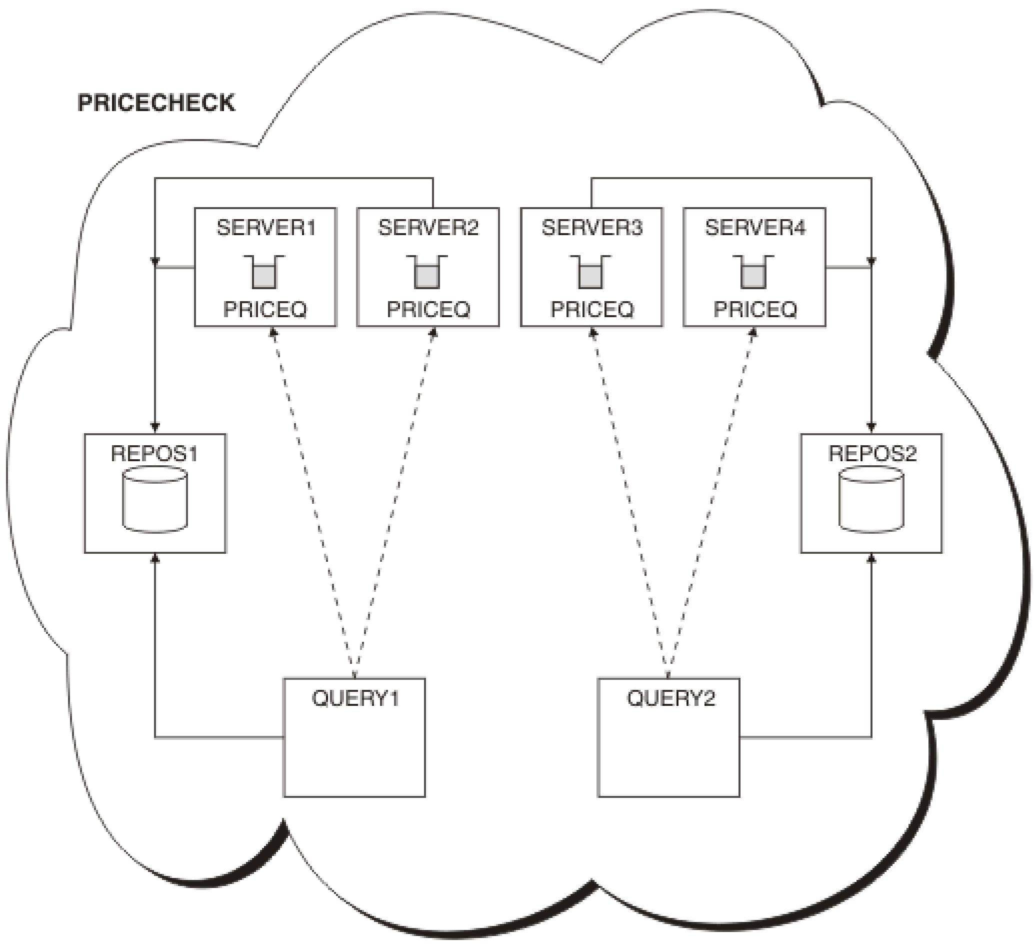

Figure 64. The PRICECHECK cluster, with four server queue managers, two repositories, and two query queue managers

Although there are four instances of the PRICEQ queue available in the PRICECHECK cluster, each querying queue manager only uses two of two of them. For example, the QUERY1 queue manager only has active channels to the SERVER1 and SERVER2 queue managers. If SERVER1 became unavailable, the QUERY1 queue manager would then begin to use another queue manager, for example SERVER3.

Related concepts

Example of a cluster with more than one instance of a queue
In this example of a cluster with more than one instance of a queue, messages are routed to different instances of the queue. You can force a message to a specific instance of the queue, and you can choose to send a sequence of messages to one of either of the queue managers.

Application programming and clusters
You do not need to make any programming changes to take advantage of multiple instances of the same queue. However, some programs do not work correctly unless a sequence of messages is sent to the same instance of a queue.

Related tasks

Adding a queue manager that hosts a queue locally
Follow these instructions to add an instance of INVENTQ to provide additional capacity to run the inventory application system in Paris and New York.

Using two networks in a cluster
Follow these instructions to add a new store in TOKYO where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

Using a primary and a secondary network in a cluster

Follow these instructions to make one network the primary network, and another network the backup network. Use the backup network if there is a problem with the primary network.

Adding a queue to act as a backup
Follow these instructions to provide a backup in Chicago for the inventory system that now runs in New York. The Chicago system is only used when there is a problem with the New York system.

Adding a more powerful queue manager that hosts a queue
Follow these instructions to provide additional capacity by running the inventory system in Los Angeles as well as New York, where Los Angeles can handle twice the number of messages as New York.

Adding a more powerful queue manager that hosts a queue

Follow these instructions to provide additional capacity by running the inventory system in Los Angeles as well as New York, where Los Angeles can handle twice the number of messages as New York.

Before you begin

Note: For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

Scenario:

- The INVENTORY cluster has been set up as described in "Adding a queue manager to a cluster" on page 237. It contains three queue managers: LONDON and NEWYORK both hold full repositories, PARIS holds a partial repository and puts messages from INVENTQ. The inventory application runs on the system in New York connected to the NEWYORK queue manager. The application is driven by the arrival of messages on the INVENTQ queue.
- A new store is being set up in Los Angeles. To provide additional capacity, you want to run the inventory system in Los Angeles as well as New York. The new queue manager can process twice as many messages as New York.

About this task

Follow these steps to add a more powerful queue manager that hosts a queue.

Procedure

1. Decide which full repository LOSANGELES refers to first.
2. Every queue manager in a cluster must refer to one or other of the full repositories to gather information about the cluster. It builds up its own partial repository. It is of no particular significance which repository you choose. In this example, NEWYORK is chosen. Once the new queue manager has joined the cluster it communicates with both of the repositories.

```
DEFINE CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(NEWYORK.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('Cluster-sender channel from LOSANGELES to repository at NEWYORK')
```

3. Define the CLUSRCVR channel on queue manager LOSANGELES.

   Every queue manager in a cluster must define a cluster-receiver channel on which it can receive messages. On LOSANGELES, define:

```
DEFINE CHANNEL(INVENTORY.LOSANGELES) CHLTYPE(CLUSRCVR) TRPTYPE(TCP)
CONNAME(LOSANGELES.CHSTORE.COM) CLUSTER(INVENTORY)
DESCR('Cluster-receiver channel for queue manager LOSANGELES')
CLWLWGHT(2)
```

   The cluster-receiver channel advertises the availability of the queue manager to receive messages from other queue managers in the cluster INVENTORY. Setting CLWLWGHT to two ensures that the Los Angeles queue manager gets twice as many of the inventory messages as New York (when the channel for NEWYORK is set to one).

4. Alter the CLUSRCVR channel on queue manager NEWYORK.

   Ensure that the Los Angeles queue manager gets twice as many of the inventory messages as New York. Alter the definition of the cluster-receiver channel.

   ALTER CHANNEL(INVENTORY.NEWYORK) CHLTYPE(CLUSRCVR) CLWLWGHT(1)
5. Review the inventory application for message affinities.

   Before proceeding, ensure that the inventory application does not have any dependencies on the sequence of processing of messages.
6. Install the inventory application on the system in Los Angeles
7. Define the cluster queue INVENTQ.

   The INVENTQ queue, which is already hosted by the NEWYORK queue manager, is also to be hosted by LOSANGELES. Define it on the LOSANGELES queue manager as follows:

   ```
   DEFINE QLOCAL(INVENTQ) CLUSTER(INVENTORY)
   ```

   Now that you have completed all the definitions, if you have not already done so start the channel initiator on IBM MQ for z/OS. On all platforms, start a listener program on queue manager LOSANGELES. The listener program listens for incoming network requests and starts the cluster-receiver channel when it is needed.

Results

"Adding a more powerful queue manager that hosts a queue" on page 325 shows the cluster set up by this task.

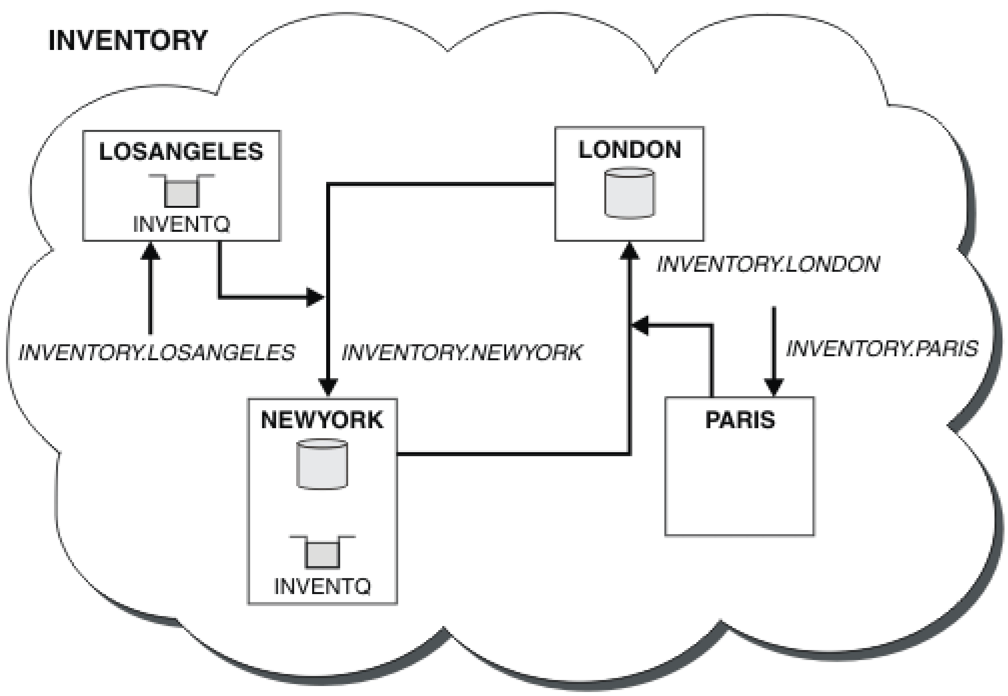

Figure 65. The INVENTORY cluster with four queue managers

This modification to the cluster was accomplished without you having to alter the queue managers LONDON and PARIS. The repositories in these queue managers are updated automatically with the information they need to be able to send messages to INVENTQ at LOSANGELES.

What to do next

The INVENTQ queue and inventory application are hosted on two queue managers in the cluster. The configuration increases their availability, speeds up throughput of messages, and allows the workload to be distributed between the two queue managers. Messages put to INVENTQ by either LOSANGELES or NEWYORK are handled by the instance on the local queue manager whenever possible. Messages put by LONDON or PARIS are routed to LOSANGELES or NEWYORK, with twice as many messages being sent to LOSANGELES.

Related concepts

Example of a cluster with more than one instance of a queue
In this example of a cluster with more than one instance of a queue, messages are routed to different instances of the queue. You can force a message to a specific instance of the queue, and you can choose to send a sequence of messages to one of either of the queue managers.

Application programming and clusters
You do not need to make any programming changes to take advantage of multiple instances of the same queue. However, some programs do not work correctly unless a sequence of messages is sent to the same instance of a queue.

Related tasks

Adding a queue manager that hosts a queue locally
Follow these instructions to add an instance of INVENTQ to provide additional capacity to run the inventory application system in Paris and New York.

Using two networks in a cluster
Follow these instructions to add a new store in TOKYO where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

Using a primary and a secondary network in a cluster
Follow these instructions to make one network the primary network, and another network the backup network. Use the backup network if there is a problem with the primary network.

Adding a queue to act as a backup
Follow these instructions to provide a backup in Chicago for the inventory system that now runs in New York. The Chicago system is only used when there is a problem with the New York system.

Restricting the number of channels used
Follow these instructions to restrict the number of active channels each server runs when a price check application is installed on various queue managers.

Application programming and clusters

You do not need to make any programming changes to take advantage of multiple instances of the same queue. However, some programs do not work correctly unless a sequence of messages is sent to the same instance of a queue.

Applications can open a queue using the MQOPEN call. Applications use the MQPUT call to put messages onto an open queue. Applications can put a single message onto a queue that is not already open, using the MQPUT1 call.

If you set up clusters that have multiple instances of the same queue, there are no specific application programming considerations. However, to benefit from the workload management aspects of clustering, you might need to modify your applications. If you set up a network in which there are multiple definitions of the same queue, review your applications for message affinities.

Suppose for example, you have two applications that rely on a series of messages flowing between them in the form of questions and answers. You probably want answers to go back to the same queue manager that sent a question. It is important that the workload management routine does not send the messages to any queue manager that hosts a copy of the reply queue.

You might have applications that require messages to be processed in sequence (for example, a database replication application that sends batches of messages that must be retrieved in sequence). The use of segmented messages can also cause an affinity problem.

Opening a local or remote version of the target queue

Be aware of how the queue manager chooses whether use a local or remote version of the target queue.

1. The queue manager opens the local version of the target queue to read messages, or to set the attributes of the queue.
2. The queue manager opens any instance of the target queue to write messages to, if at least one of the following conditions is true:
   - A local version of the target queue does not exist.
   - The queue manager specifies `CLWLUSEQ(ANY)` on `ALTER QMGR`.
   - The queue on the queue manager specifies `CLWLUSEQ(ANY)`.

Related concepts

Example of a cluster with more than one instance of a queue
In this example of a cluster with more than one instance of a queue, messages are routed to different instances of the queue. You can force a message to a specific instance of the queue, and you can choose to send a sequence of messages to one of either of the queue managers.

Related tasks

Adding a queue manager that hosts a queue locally
Follow these instructions to add an instance of `INVENTQ` to provide additional capacity to run the inventory application system in Paris and New York.

Using two networks in a cluster
Follow these instructions to add a new store in `TOKYO` where there are two different networks. Both need to be available for use to communicate with the queue manager in Tokyo.

Using a primary and a secondary network in a cluster
Follow these instructions to make one network the primary network, and another network the backup network. Use the backup network if there is a problem with the primary network.

Adding a queue to act as a backup
Follow these instructions to provide a backup in Chicago for the inventory system that now runs in New York. The Chicago system is only used when there is a problem with the New York system.

Restricting the number of channels used
Follow these instructions to restrict the number of active channels each server runs when a price check application is installed on various queue managers.

Adding a more powerful queue manager that hosts a queue
Follow these instructions to provide additional capacity by running the inventory system in Los Angeles as well as New York, where Los Angeles can handle twice the number of messages as New York.

Handling message affinities
Message affinities are rarely part of good programming design. You need to remove message affinities to use clustering fully. If you cannot remove message affinities, you can force related messages to be delivered using the same channel and to the same queue manager.

If you have applications with message affinities, remove the affinities before starting to use clusters.

Removing message affinities improves the availability of applications. An application sends a batch of messages that has message affinities to a queue manager. The queue manager fails after receiving only part of the batch. The sending queue manager must wait for it to recover and process the incomplete message batch before it can send any more messages.

Removing messages affinities also improves the scalability of applications. A batch of messages with affinities can lock resources at the destination queue manager while waiting for subsequent messages. These resources might remain locked for long periods of time, preventing other applications from doing their work.

Furthermore, message affinities prevent the cluster workload management routines from making the best choice of queue manager.

To remove affinities, consider the following possibilities:

- Carrying state information in the messages
- Maintaining state information in nonvolatile storage accessible to any queue manager, for example in a Db2 database
- Replicating read-only data so that it is accessible to more than one queue manager

If it is not appropriate to modify your applications to remove message affinities, there are a number of possible solutions to the problem.

Name a specific destination on the MQOPEN call

Specify the remote-queue name and the queue manager name on each MQOPEN call, and all messages put to the queue using that object handle go to the same queue manager, which might be the local queue manager.

Specifying the remote-queue name and the queue manager name on each MQOPEN call has disadvantages:

- No workload balancing is carried out. You do not take advantage of the benefits of cluster workload balancing.
- If the target queue manager is remote and there is more than one channel to it, the messages might take different routes and the sequence of messages is still not preserved.
- If your queue manager has a definition for a transmission queue with the same name as the destination queue manager, messages go on that transmission queue rather than on the cluster transmission queue.

Return the queue-manager name in the reply-to queue manager field

Allow the queue manager that receives the first message in a batch to return its name in its response. It does this using the `ReplyToQMgr` field of the message descriptor. The queue manager at the sending end can then extract the reply-to queue manager name and specify it on all subsequent messages.

Using the `ReplyToQMgr` information from the response has disadvantages:

- The requesting queue manager must wait for a response to its first message
- You must write additional code to find and use the `ReplyToQMgr` information before sending subsequent messages
- If there is more than one route to the queue manager, the sequence of the messages might not be preserved

Set the MQOO_BIND_ON_OPEN option on the MQOPEN call

Force all your messages to be put to the same destination using the MQOO_BIND_ON_OPEN option on the MQOPEN call. Either MQOO_BIND_ON_OPEN or MQOO_BIND_ON_GROUP must be specified when using message groups with clusters to ensure that all messages in the group are processed at the same destination.

By opening a queue and specifying MQOO_BIND_ON_OPEN, you force all messages that are sent to this queue to be sent to the same instance of the queue. MQOO_BIND_ON_OPEN binds all messages to the same queue manager and also to the same route. For example, if there is an IP route and a NetBIOS route to the same destination, one of these is selected when the queue is opened and this selection is honored for all messages put to the same queue using the object handle obtained.

By specifying MQOO_BIND_ON_OPEN you force all messages to be routed to the same destination. Therefore applications with message affinities are not disrupted. If the destination is not available, the messages remain on the transmission queue until it becomes available again.

MQOO_BIND_ON_OPEN also applies when the queue manager name is specified in the object descriptor when you open a queue. There might be more than one route to the named queue manager. For example, there might be multiple network paths or another queue manager might have defined an alias. If you specify MQOO_BIND_ON_OPEN, a route is selected when the queue is opened.

Note: This is the recommended technique. However, it does not work in a multi-hop configuration in which a queue manager advertises an alias for a cluster queue. Nor does it help in situations in which applications use different queues on the same queue manager for different groups of messages.

An alternative to specifying MQOO_BIND_ON_OPEN on the MQOPEN call, is to modify your queue definitions. On your queue definitions, specify DEFBIND(OPEN), and allow the DefBind option on the MQOPEN call to default to MQOO_BIND_AS_Q_DEF.

Set the MQOO_BIND_ON_GROUP option on the MQOPEN call

Force all your messages in a group to be put to the same destination using the MQOO_BIND_ON_GROUP option on the MQOPEN call. Either MQOO_BIND_ON_OPEN or MQOO_BIND_ON_GROUP must be specified when using message groups with clusters to ensure that all messages in the group are processed at the same destination.

By opening a queue and specifying MQOO_BIND_ON_GROUP, you force all messages in a group that are sent to this queue to be sent to the same instance of the queue. MQOO_BIND_ON_GROUP binds all messages in a group to the same queue manager, and also to the same route. For example, if there is an IP route and a NetBIOS route to the same destination, one of these is selected when the queue is opened and this selection is honored for all messages in a group put to the same queue using the object handle obtained.

By specifying MQOO_BIND_ON_GROUP you force all messages in a group to be routed to the same destination. Therefore applications with message affinities are not disrupted. If the destination is not available, the messages remain on the transmission queue until it becomes available again.

MQOO_BIND_ON_GROUP also applies when the queue manager name is specified in the object descriptor when you open a queue. There might be more than one route to the named queue manager. For example, there might be multiple network paths or another queue manager might have defined an alias. If you specify MQOO_BIND_ON_GROUP, a route is selected when the queue is opened.

For MQOO_BIND_ON_GROUP to be effective you must include the MQPMO_LOGICAL_ORDER put option on MQPUT. You can set **GroupId** in the MQMD of the message to MQGI_NONE, and you must include the following message flags within the MQMD **MsgFlags** field of the messages:

- Last message in group: MQMF_LAST_MSG_IN_GROUP
- All other messages in group: MQMF_MSG_IN_GROUP

If MQOO_BIND_ON_GROUP is specified but the messages are not grouped, the behavior is equivalent to MQOO_BIND_NOT_FIXED.

Note: This is the recommended technique for ensuring that messages in a group are sent to the same destination. However, it does not work in a multi-hop configuration in which a queue manager advertises an alias for a cluster queue.

An alternative to specifying MQOO_BIND_ON_GROUP on the MQOPEN call, is to modify your queue definitions. On your queue definitions, specify DEFBIND(GROUP), and allow the DefBind option on the MQOPEN call to default to MQOO_BIND_AS_Q_DEF.

Write a customized cluster workload exit program

Instead of modifying your applications you can circumvent the message affinities problem by writing a cluster workload exit program. Writing a cluster workload exit program is not easy and is not a recommended solution. The program would have to be designed to recognize the affinity by inspecting the content of messages. Having recognized the affinity, the program would have to force the workload management utility to route all related messages to the same queue manager.

Configuring publish/subscribe messaging

You can start, stop and display the status of queued publish/subscribe. You can also add and remove streams, and add and delete queue managers from a broker hierarchy.

See the following subtopics for more information on controlling queued publish/subscribe:

Setting queued publish/subscribe message attributes

You control the behavior of some publish/subscribe message attributes using queue manager attributes. The other attributes you control in the *Broker* stanza of the `qm.ini` file.

About this task

You can set the following publish/subscribe attributes: for details see, Queue manager parameters

Table 26. Publish/subscribe configuration parameters

Description	MQSC parameter name
Command message retry count	**PSRTYCNT**
Discard undeliverable command input message	**PSNPMSG**
Behavior following undeliverable command response message	**PSNPRES**
Process command messages under syncpoint	**PSSYNCPT**

The Broker stanza is used to manage the following configuration settings:

- `PersistentPublishRetry=yes | force`

  If you specify `Yes`, then if a publication of a persistent message through the queued publish/subscribe interface fails, and no negative reply was requested, the publish operation is retried.

  If you requested a negative response message, the negative response is sent and no further retry occurs.

  If you specify `Force`, then if a publication of a persistent message through the queued publish/subscribe interface fails, the publish operation is retried until the it is successfully processed. No negative response is sent.

- `NonPersistentPublishRetry=yes | force`

  If you specify `Yes`, then if a publication of a non-persistent message through the queued publish/subscribe interface fails, and no negative reply was requested, the publish operation is retried.

  If you requested a negative response message, the negative response is sent and no further retry occurs.

  If you specified `Force`, then if a publication of a non-persistent message through the queued publish/subscribe interface fails, the publish operation is retried until it is successfully processed. No negative response is sent.

  Note: If you want to enable this functionality for non-persistent messages, then as well as setting the `NonPersistentPublishRetry` value you must also ensure that the queue manager attribute **PSSYNCPT** is set to `Yes`.

  Doing this might also have an impact on the performance of processing non-persistent publications as the **MQGET** from the STREAM queue now occurs under syncpoint.

- `PublishBatchSize=`*number*

  The broker normally processes publish messages within syncpoint. It can be inefficient to commit each publication individually, and in some circumstances the broker can process multiple publish messages in a single unit of work. This parameter specifies the maximum number of publish messages that can be processed in a single unit of work

The default value for `PublishBatchSize` is 5.

- `PublishBatchInterval=`*number*

  The broker normally processes publish messages within syncpoint. It can be inefficient to commit each publication individually, and in some circumstances the broker can process multiple publish messages in a single unit of work. This parameter specifies the maximum time (in milliseconds) between the first message in a batch and any subsequent publication included in the same batch.

  A batch interval of 0 indicates that up to `PublishBatchSize` messages can be processed, provided that the messages are available immediately.

  The default value for `PublishBatchInterval` is zero.

Procedure

Use IBM MQ Explorer, programmable commands, or the **`runmqsc`** command to alter the queue manager attributes that control the behavior of publish/subscribe.

Example

ALTER QMGR PSNPRES(SAFE)

Starting queued publish/subscribe

Before you begin

Read the description of PSMODE to understand the three modes of publish/subscribe:

- `COMPAT`
- `DISABLED`
- `ENABLED`

About this task

Set the QMGR `PSMODE` attribute to start either the queued publish/subscribe interface (also known as the broker), or the publish/subscribe engine (also known as Version 7 publish/subscribe) or both. To start queued publish/subscribe you need to set `PSMODE` to `ENABLED`. The default is `ENABLED`.

Procedure

Use IBM MQ Explorer or the **`runmqsc`** command to enable the queued publish/subscribe interface if the interface is not already enabled.

Example

```
ALTER QMGR PSMODE (ENABLED)
```

What to do next

IBM MQ processes queued publish/subscribe commands and publish/subscribe Message Queue Interface (MQI) calls.

Stopping queued publish/subscribe

Before you begin

Read the description of PSMODE to understand the three modes of publish/subscribe:

- COMPAT
- DISABLED
- ENABLED

About this task

Set the QMGR PSMODE attribute to stop either the queued publish/subscribe interface (also known as the broker), or the publish/subscribe engine (also known as Version 7 publish/subscribe) or both. To stop queued publish/subscribe you need to set PSMODE to `COMPAT`. To stop the publish/subscribe engine entirely, set PSMODE to `DISABLED`.

Procedure

Use IBM MQ Explorer or the **`runmqsc`** command to disable the queued publish/subscribe interface.

Example

```
ALTER QMGR PSMODE (COMPAT)
```

Adding a stream

You can add streams manually to allow for data isolation between applications, or to allow inter-operation with Version 6 publish/subscribe hierarchies.

Before you begin

Familiarize yourself with the way publish/subscribe streams operate. See Streams and topics.

About this task

Use PCF command, **`runmqsc`**, or MQ Explorer to do these steps.

Note: You can perform steps 1 and 2 in any order. Only perform step 3 after steps 1 and 2 have both been completed.

Procedure

1. Define a local queue with the same name as the Version 6 stream.
2. Define a local topic with the same name as the Version 6 stream.
3. Add the name of the queue to the namelist, `SYSTEM.QPUBSUB.QUEUE.NAMELIST`
4. Repeat for all queue managers at Version 7.1 or above that are in the publish/subscribe hierarchy.

Adding `'Sport'`

In the example of sharing the stream `'Sport'`, Version 6 and Version 7.1 queue managers are working in the same publish/subscribe hierarchy. The Version 6 queue managers share a stream called `'Sport'`. The example shows how to create a queue and a topic on Version 7.1 queue managers called `'Sport'`, with a topic string `'Sport'` that is shared with the Version 6 stream `'Sport'`.

A Version 7.1 publish application, publishing to topic `'Sport'`, with topic string `'Soccer/Results'`, creates the resultant topic string `'Sport/Soccer/Results'`. On Version 7.1 queue managers, subscribers to topic `'Sport'`, with topic string `'Soccer/Results'` receive the publication.

On Version 6 queue managers, subscribers to stream `'Sport'`, with topic string `'Soccer/Results'` receive the publication.

```
runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.
define qlocal('Sport')
     1 : define qlocal('Sport')
AMQ8006: IBM MQ queue created.
define topic('Sport') topicstr('Sport')
     2 : define topic('Sport') topicstr('Sport')
AMQ8690: IBM MQ topic created.
alter namelist(SYSTEM.QPUBSUB.QUEUE.NAMELIST) NAMES('Sport', 'SYSTEM.BROKER.DEFAULT.STREAM',
'SYSTEM.BROKER.ADMIN.STREAM')
     3 : alter namelist(SYSTEM.QPUBSUB.QUEUE.NAMELIST) NAMES('Sport', 'SYSTEM.BROKER.DEFAULT.STREAM',
'SYSTEM.BROKER.ADMIN.STREAM')
AMQ8551: IBM MQ namelist changed.
```

Note: You need both to provide the existing names in the namelist object, as well as the new names that you are adding, to the **`alter namelist`** command.

What to do next

Information about the stream is passed to other brokers in the hierarchy.

If a broker is Version 6, administer it as a Version 6 broker. That is, you have a choice of creating the stream queue manually, or letting the broker create the stream queue dynamically when it is needed. The queue is based on the model queue definition, `SYSTEM.BROKER.MODEL.STREAM`.

If a broker is Version 7.1, you must configure each Version 7.1 queue manager in the hierarchy manually.

Deleting a stream

You can delete a stream from an IBM MQ Version 7.1, or later, queue manager.

Before you begin

Before deleting a stream you must ensure that there are no remaining subscriptions to the stream and quiesce all applications that use the stream. If publications continue to flow to a deleted stream, it takes a lot of administrative effort to restore the system to a cleanly working state.

Procedure

1. Find all the connected brokers that host this stream.
2. Cancel all subscriptions to the stream on all the brokers.
3. Remove the queue (with the same name as the stream) from the namelist, `SYSTEM.QPUBSUB.QUEUE.NAMELIST`.
4. Delete or purge all the messages from the queue with the same name as the stream.
5. Delete the queue with the same name as the stream.
6. Delete the associated topic object.

What to do next

Repeat steps 3 to 5 on all the other connected Version 7.1, or later, queue managers hosting the stream.

Adding a subscription point

How to extend an existing queued publish/subscribe application that you have migrated from an earlier version of IBM Integration Bus with a new subscription point.

Before you begin

1. Check that the subscription point is not already defined in `SYSTEM.QPUBSUB.SUBPOINT.NAMELIST`.

2. Check if there is a topic object or a topic string with the same name as the subscription point.

About this task

IBM WebSphere MQ 7.1, or later, applications do not use subscription points, but they can interoperate with existing applications that do, using the subscription point migration mechanism.

Important: The subscription point migration mechanism has been removed from IBM MQ 8.0. If you need to migrate your existing applications, you must carry out the procedures described in the documentation for your version of the product, before you migrate to the latest version.

Subscription points do not work with queued publish/subscribe programs that use `MQRFH1` headers, which have been migrated from IBM MQ Version 6, or earlier.

There is no need to add subscription points to use integrated publish/subscribe applications written for IBM MQ Version 7.1, or later.

Procedure

1. Add the name of the subscription point to `SYSTEM.QPUBSUB.SUBPOINT.NAMELIST`.
   - On z/OS, the **NLTYPE** is `NONE`, the default.
   - Repeat the step on every queue manager that is connected in the same publish/subscribe topology.
2. Add a topic object, preferably giving it the name of the subscription point, with a topic string matching the name of the subscription point.
   - If the subscription point is in a cluster, add the topic object as a cluster topic on the cluster topic host.
   - If a topic object exists with the same topic string as the name of the subscription point, use the existing topic object. You must understand the consequences of the subscription point reusing an existing topic. If the existing topic is part of an existing application, you must resolve the collision between two identically named topics.
   - If a topic object exists with the same name as the subscription point, but a different topic string, create a topic with a different name.
3. Set the **`Topic`** attribute `WILDCARD` to the value `BLOCK`.

   Blocking subscriptions to # or * isolates wildcard subscriptions to subscription points, see Wildcards and subscription points.
4. Set any attributes that you require in the topic object.

Example

The example shows a **`runmqsc`** command file that adds two subscription points, USD and GBP.

```
DEFINE TOPIC(USD) TOPICSTR(USD)
DEFINE TOPIC(GBP) TOPICSTR(GBP) WILDCARD(BLOCK)
ALTER NL(SYSTEM.QPUBSUB.SUBPOINT.NAMELIST) NAMES(SYSTEM.BROKER.DEFAULT.SUBPOINT, USD, GBP)
```

Note:

1. Include the default subscription point in the list of subscription points added using the **ALTER** command. **ALTER** deletes existing names in the namelist.
2. Define the topics before altering the namelist. The queue manager only checks the namelist when the queue manager starts and when the namelist is altered.

Configuring distributed publish/subscribe networks

Queue managers that are connected together into a distributed publish/subscribe topology share a common federated topic space. Subscriptions created on one queue manager can receive messages published by an application connected to another queue manager in the topology.

You can control the extent of topic spaces created by connecting queue managers together in clusters or hierarchies. In a publish/subscribe cluster, a topic object must be 'clustered' for each branch of the topic space that is to span the cluster. In a hierarchy, each queue manager must be configured to identify its 'parent' in the hierarchy.

You can further control the flow of publications and subscriptions within the topology by choosing whether each publication and subscription is either local or global. Local publications and subscriptions are not propagated beyond the queue manager to which the publisher or subscriber is connected.

Related concepts

Distributed publish/subscribe networks

Publication scope

Subscription scope

Topic spaces

Related tasks

Defining cluster topics

Configuring a publish/subscribe cluster

Define a topic on a queue manager. To make the topic a cluster topic, set the **CLUSTER** property. To choose the routing to use for publications and subscriptions for this topic, set the **CLROUTE** property.

Before you begin

Some cluster configurations cannot accommodate the overheads of direct routed publish/subscribe. Before you use this configuration, explore the considerations and options detailed in Designing publish/subscribe clusters.

For changes to a cluster to be propagated throughout the cluster, at least one full repository must always be available. Ensure that your repositories are available before starting this task.

See also Routing for publish/subscribe clusters: Notes on behavior.

Scenario:

- The INVENTORY cluster has been set up as described in "Adding a queue manager to a cluster" on page 237. It contains three queue managers; LONDON and NEWYORK both hold full repositories, PARIS holds a partial repository.

About this task

When you define a topic on a queue manager in a cluster, you need to specify whether the topic is a cluster topic, and (if so) the routing within the cluster for publications and subscriptions for this topic. To make the topic a cluster topic, you configure the **CLUSTER** property on the TOPIC object with the name of the cluster. By defining a cluster topic on a queue manager in the cluster, you make the topic available to the whole cluster. To choose the message routing to use within the cluster, you set the **CLROUTE** property on the TOPIC object to one of the following values:

- **DIRECT**
- **TOPICHOST**

By default, topic routing is **DIRECT**. This was the only option prior to IBM MQ 8.0. When you configure a direct routed clustered topic on a queue manager, all queue managers in the cluster become aware of all other queue managers in the cluster. When performing publish and subscribe operations, each queue manager can connect direct to any other queue manager in the cluster. See Direct routed publish/subscribe clusters.

From IBM MQ 8.0, you can instead configure topic routing as **TOPICHOST**. When you use topic host routing, all queue managers in the cluster become aware of the cluster queue managers that host the routed topic definition (that is, the queue managers on which you have defined the topic object). When performing publish and subscribe operations, queue managers in the cluster connect only to these topic host queue managers, and not directly to each other. The topic host queue managers are responsible for routing publications from queue managers on which publications are published to queue managers with matching subscriptions. See Topic host routed publish/subscribe clusters.

Note: After a topic object has been clustered (through setting the **CLUSTER** property) you cannot change the value of the **CLROUTE** property. The object must be un-clustered (**CLUSTER** set to `' '`) before you can change the value. Un-clustering a topic converts the topic definition to a local topic, which results in a period during which publications are not delivered to subscriptions on remote queue managers; this should be considered when performing this change. See The effect of defining a non-cluster topic with the same name as a cluster topic from another queue manager. If you try to change the value of the **CLROUTE** property while it is clustered, the system generates an `MQRCCF_CLROUTE_NOT_ALTERABLE` exception.

Procedure

1. Choose a queue manager to host your topic.

   Any cluster queue manager can host a topic. Choose one of the three queue managers (`LONDON`, `NEWYORK` or `PARIS`) and configure the properties of the `TOPIC` object. If you plan to use direct routing, it makes no operational difference which queue manager you choose. If you plan to use topic host routing, the chosen queue manager has additional responsibilities for routing publications. Therefore, for topic host routing, choose a queue manager that is hosted on one of your more powerful systems and has good network connectivity.
2. Define a topic on a queue manager.

   To make the topic a cluster topic, include the cluster name when you define the topic, and set the routing that you want to use for publications and subscriptions for this topic. For example, to create a direct routing cluster topic on the `LONDON` queue manager, create the topic as follows:

   ```
   DEFINE TOPIC(INVENTORY) TOPICSTR('/INVENTORY') CLUSTER(INVENTORY) CLROUTE(DIRECT)
   ```

   By defining a cluster topic on a queue manager in the cluster, you make the topic available to the whole cluster.

   For more information about using **CLROUTE**, see DEFINE TOPIC (CLROUTE) and Routing for publish/subscribe clusters: Notes on behavior.

Results

The cluster is ready to receive publications and subscriptions for the topic.

What to do next

If you have configured a topic host routed publish/subscribe cluster, you will probably want to add a second topic host for this topic. See "Adding extra topic hosts to a topic host routed cluster" on page 339.

If you have several separate publish/subscribe clusters, for example because your organization is geographically dispersed, you might want to propagate some cluster topics into all the clusters. You can do this by connecting the clusters in a hierarchy. See "Combining the topic spaces of multiple clusters" on page 345. You can also control which publications flow from one cluster to another. See "Combining and isolating topic spaces in multiple clusters" on page 347.

Related concepts

Combining publication and subscription scopes
In IBM MQ versions 7 onwards, publication and subscription scope work independently to determine the flow of publications between queue managers.

Combining topic spaces in publish/subscribe networks

Combine the topic space of a queue manager with other queue managers in a publish/subscribe cluster or hierarchy. Combine publish/subscribe clusters, and publish/subscribe clusters with hierarchies.

Related tasks

Moving a cluster topic definition to a different queue manager
For either topic host routed or direct routed clusters, you might need to move a cluster topic definition when decommissioning a queue manager, or because a cluster queue manager has failed or is unavailable for a significant period of time.

Adding extra topic hosts to a topic host routed cluster
In a topic host routed publish/subscribe cluster, multiple queue managers can be used to route publications to subscriptions by defining the same clustered topic object on those queue managers. This can be used to improve availability and workload balancing. When you add an extra topic host for the same cluster topic object, you can use the **PUB** parameter to control when publications begin to be routed through the new topic host.

Connecting a queue manager to a publish/subscribe hierarchy
You connect the child queue manager to the parent queue manager in the hierarchy. If the child queue manager is already a member of another hierarchy or cluster, then this connection joins the hierarchies together, or joins the cluster to the hierarchy.

Disconnecting a queue manager from a publish/subscribe hierarchy
Disconnect a child queue manager from a parent queue manager in a publish/subscribe hierarchy.

Designing publish/subscribe clusters

Distributed publish/subscribe troubleshooting

Inhibiting clustered publish/subscribe

Moving a cluster topic definition to a different queue manager

For either topic host routed or direct routed clusters, you might need to move a cluster topic definition when decommissioning a queue manager, or because a cluster queue manager has failed or is unavailable for a significant period of time.

About this task

You can have multiple definitions of the same cluster topic object in a cluster. This is a normal state for a topic host routed cluster, and an unusual state for a direct routed cluster. For more information, see Multiple cluster topic definitions of the same name.

To move a cluster topic definition to a different queue manager in the cluster without interrupting the flow of publications, complete the following steps. The procedure moves a definition from queue manager `QM1` to queue manager `QM2`.

Procedure

1. Create a duplicate of the cluster topic definition on `QM2`.

   For direct routing, set all the attributes to match the definition of `QM1`.

   For topic host routing, initially define the new topic host as `PUB(DISABLED)`. This allows `QM2` to learn of the subscriptions in the cluster, but not to start routing publications.
2. Wait for information to be propagated through the cluster.

   Wait for the new cluster topic definition to be propagated by the full repository queue managers to all queue managers in the cluster. Use the **DISPLAY CLUSTER** command to display the cluster topics on each cluster member, and check for a definition originating from `QM2`.

For topic host routing, wait for the new topic host on QM2 to learn of all subscriptions. Compare the proxy subscriptions known to QM2 and those known to QM1. One way to view the proxy subscriptions on a queue manager is to issue the following **runmqsc** command:

```
DISPLAY SUB(*) SUBTYPE(PROXY)
```

3. For topic host routing, redefine the topic host on QM2 as PUB(ENABLED), then redefine the topic host on QM1 as PUB(DISABLED).

   Now that the new topic host on QM2 has learned of all subscriptions on other queue managers, the topic host can start routing publications.

   By using the PUB(DISABLED) setting to quiesce message traffic through QM1, you ensure that no publications are in train through QM1 when you delete the cluster topic definition.
4. Delete the cluster topic definition from QM1.

   You can only delete the definition from QM1 if the queue manager is available. Otherwise, you must run with both definitions in existence until QM1 is restarted or forcibly removed.

   If QM1 remains unavailable for a long time, and during that time you need to modify the clustered topic definition on QM2, the QM2 definition is newer than the QM1 definition, and therefore usually prevails.

   During this period, if there are differences between the definitions on QM1 and QM2, errors are written to the error logs of both queue managers, alerting you to the conflicting cluster topic definition.

   If QM1 is never going to return to the cluster, for example because of unexpected decommissioning following a hardware failure, as a last resort you can use the RESET CLUSTER command to forcibly eject the queue manager. **RESET CLUSTER** automatically deletes all topic objects hosted on the target queue manager.

Related concepts

Combining publication and subscription scopes
In IBM MQ versions 7 onwards, publication and subscription scope work independently to determine the flow of publications between queue managers.

Combining topic spaces in publish/subscribe networks
Combine the topic space of a queue manager with other queue managers in a publish/subscribe cluster or hierarchy. Combine publish/subscribe clusters, and publish/subscribe clusters with hierarchies.

Related tasks

Configuring a publish/subscribe cluster
Define a topic on a queue manager. To make the topic a cluster topic, set the **CLUSTER** property. To choose the routing to use for publications and subscriptions for this topic, set the **CLROUTE** property.

Adding extra topic hosts to a topic host routed cluster
In a topic host routed publish/subscribe cluster, multiple queue managers can be used to route publications to subscriptions by defining the same clustered topic object on those queue managers. This can be used to improve availability and workload balancing. When you add an extra topic host for the same cluster topic object, you can use the **PUB** parameter to control when publications begin to be routed through the new topic host.

Connecting a queue manager to a publish/subscribe hierarchy
You connect the child queue manager to the parent queue manager in the hierarchy. If the child queue manager is already a member of another hierarchy or cluster, then this connection joins the hierarchies together, or joins the cluster to the hierarchy.

Disconnecting a queue manager from a publish/subscribe hierarchy
Disconnect a child queue manager from a parent queue manager in a publish/subscribe hierarchy.

Adding extra topic hosts to a topic host routed cluster

In a topic host routed publish/subscribe cluster, multiple queue managers can be used to route publications to subscriptions by defining the same clustered topic object on those queue managers. This can be used to improve availability and workload balancing. When you add an extra topic host for the same cluster topic object, you can use the **PUB** parameter to control when publications begin to be routed through the new topic host.

Before you begin

Defining the same cluster topic object on several queue managers is only functionally useful for a topic host routed cluster. Defining multiple matching topics in a direct routed cluster does not change its behavior. This task only applies to topic host routed clusters.

This task assumes that you have read the article Multiple cluster topic definitions of the same name, especially the following sections:

- Multiple cluster topic definitions in a topic host routed cluster
- Special handling for the PUB parameter

About this task

When a queue manager is made a routed topic host, it must first learn of the existence of all related topics that have been subscribed to in the cluster. If publications are being published to those topics at the time that an additional topic host is added, and a publication is routed to the new host before that host has learned of the existence of subscriptions on other queue managers in the cluster, then the new host does not forward that publication to those subscriptions. This causes subscriptions to miss publications.

Publications are not routed through topic host queue managers that have explicitly set the cluster topic object **PUB** parameter to `DISABLED`, so you can use this setting to ensure that no subscriptions miss publications during the process of adding an extra topic host.

Note: While a queue manager hosts a cluster topic that has been defined as `PUB(DISABLED)`, publishers connected to that queue manager cannot publish messages, and matching subscriptions on that queue manager do not receive publications published on other queue managers in the cluster. For this reason, careful consideration must be given to defining topic host routed topics on queue managers where subscriptions exist and publishing applications connect.

Procedure

1. Configure a new topic host, and initially define the new topic host as `PUB(DISABLED)`.

   This allows the new topic host to learn of the subscriptions in the cluster, but not to start routing publications.

   For information about configuring a topic host, see "Configuring a publish/subscribe cluster" on page 336.
2. Determine when the new topic host has learned of all subscriptions.

   To do this, compare the proxy subscriptions known to the new topic host and those known to the existing topic host. One way to view the proxy subscriptions is to issue the following **runmqsc** command: `DISPLAY SUB(*) SUBTYPE(PROXY)`
3. Redefine the new topic host as `PUB(ENABLED)`.

   After the new topic host has learned of all subscriptions on other queue managers, the topic can start routing publications.

Related concepts

Combining publication and subscription scopes
In IBM MQ versions 7 onwards, publication and subscription scope work independently to determine the flow of publications between queue managers.

Combining topic spaces in publish/subscribe networks

Combine the topic space of a queue manager with other queue managers in a publish/subscribe cluster or hierarchy. Combine publish/subscribe clusters, and publish/subscribe clusters with hierarchies.

Related tasks

Configuring a publish/subscribe cluster
Define a topic on a queue manager. To make the topic a cluster topic, set the **CLUSTER** property. To choose the routing to use for publications and subscriptions for this topic, set the **CLROUTE** property.

Moving a cluster topic definition to a different queue manager
For either topic host routed or direct routed clusters, you might need to move a cluster topic definition when decommissioning a queue manager, or because a cluster queue manager has failed or is unavailable for a significant period of time.

Connecting a queue manager to a publish/subscribe hierarchy
You connect the child queue manager to the parent queue manager in the hierarchy. If the child queue manager is already a member of another hierarchy or cluster, then this connection joins the hierarchies together, or joins the cluster to the hierarchy.

Disconnecting a queue manager from a publish/subscribe hierarchy
Disconnect a child queue manager from a parent queue manager in a publish/subscribe hierarchy.

Combining publication and subscription scopes

In IBM MQ versions 7 onwards, publication and subscription scope work independently to determine the flow of publications between queue managers.

Publications can flow to all queue managers that are connected in a publish/subscribe topology, or only to the local queue manager. Similarly for proxy subscriptions. Which publications match a subscription is governed by the combination of these two flows.

Publications and subscriptions can both be scoped to `QMGR` or `ALL`. If a publisher and a subscriber are both connected to the same queue manager, scope settings do not affect which publications the subscriber receives from that publisher.

If the publisher and subscriber are connected to different queue managers, both settings must be `ALL` to receive remote publications.

Suppose publishers are connected to different queue managers. If you want a subscriber to receive publications from any publisher, set the subscription scope to `ALL`. You can then decide, for each publisher, whether to limit the scope of its publications to subscribers local to the publisher.

Suppose subscribers are connected to different queue managers. If you want the publications from a publisher to be sent to all the subscribers, set the publication scope to `ALL`. If you want a subscriber to receive publications only from a publisher connected to the same queue manager, set the subscription scope to `QMGR`.

Example: football results service

Suppose you are a member team in a football league. Each team has a queue manager connected to all the other teams in a publish/subscribe cluster.

The teams publish the results of all the games played on their home ground using the topic, `Football/result/`*`Home team name/Away team name`*. The strings in italics are variable topic names, and the publication is the result of the match.

Each club also republishes the results just for the club using the topic string `Football/myteam/`*`Home team name/Away team name`*.

Both topics are published to the whole cluster.

The following subscriptions have been set up by the league so that fans of any team can subscribe to the results in three interesting ways.

Notice that you can set up cluster topics with SUBSCOPE(QMGR). The topic definitions are propagated to each member of the cluster, but the scope of the subscription is just the local queue manager. Thus subscribers at each queue manager receive different publications from the same subscription.

Receive all results

```
DEFINE TOPIC(A) TOPICSTR('Football/result/') CLUSTER SUBSCOPE(ALL)
```

Receive all home results

```
DEFINE TOPIC(B) TOPICSTR('Football/result/') CLUSTER SUBSCOPE(QMGR)
```

Because the subscription has QMGR scope, only results published at the home ground are matched.

Receive all my teams results

```
DEFINE TOPIC(C) TOPICSTR('Football/myteam/') CLUSTER SUBSCOPE(QMGR)
```

Because the subscription has QMGR scope, only the local team results, which are republished locally, are matched.

Related concepts

Combining topic spaces in publish/subscribe networks
Combine the topic space of a queue manager with other queue managers in a publish/subscribe cluster or hierarchy. Combine publish/subscribe clusters, and publish/subscribe clusters with hierarchies.

Distributed publish/subscribe networks

Publication scope

Subscription scope

Related tasks

Configuring a publish/subscribe cluster
Define a topic on a queue manager. To make the topic a cluster topic, set the **CLUSTER** property. To choose the routing to use for publications and subscriptions for this topic, set the **CLROUTE** property.

Moving a cluster topic definition to a different queue manager
For either topic host routed or direct routed clusters, you might need to move a cluster topic definition when decommissioning a queue manager, or because a cluster queue manager has failed or is unavailable for a significant period of time.

Adding extra topic hosts to a topic host routed cluster
In a topic host routed publish/subscribe cluster, multiple queue managers can be used to route publications to subscriptions by defining the same clustered topic object on those queue managers. This can be used to improve availability and workload balancing. When you add an extra topic host for the same cluster topic object, you can use the **PUB** parameter to control when publications begin to be routed through the new topic host.

Connecting a queue manager to a publish/subscribe hierarchy
You connect the child queue manager to the parent queue manager in the hierarchy. If the child queue manager is already a member of another hierarchy or cluster, then this connection joins the hierarchies together, or joins the cluster to the hierarchy.

Disconnecting a queue manager from a publish/subscribe hierarchy
Disconnect a child queue manager from a parent queue manager in a publish/subscribe hierarchy.

Combining topic spaces in publish/subscribe networks

Combine the topic space of a queue manager with other queue managers in a publish/subscribe cluster or hierarchy. Combine publish/subscribe clusters, and publish/subscribe clusters with hierarchies.

You can create different publish/subscribe topic spaces by using the building blocks of **CLUSTER**, **PUBSCOPE** and **SUBSCOPE** attributes, publish/subscribe clusters, and publish/subscribe hierarchies.

Starting from the example of scaling up from a single queue manager to a publish/subscribe cluster, the following scenarios illustrate different publish/subscribe topologies.

Related concepts

Combining publication and subscription scopes
In IBM MQ versions 7 onwards, publication and subscription scope work independently to determine the flow of publications between queue managers.

Distributed publish/subscribe networks

Topic spaces

Related tasks

Configuring a publish/subscribe cluster
Define a topic on a queue manager. To make the topic a cluster topic, set the **CLUSTER** property. To choose the routing to use for publications and subscriptions for this topic, set the **CLROUTE** property.

Moving a cluster topic definition to a different queue manager
For either topic host routed or direct routed clusters, you might need to move a cluster topic definition when decommissioning a queue manager, or because a cluster queue manager has failed or is unavailable for a significant period of time.

Adding extra topic hosts to a topic host routed cluster
In a topic host routed publish/subscribe cluster, multiple queue managers can be used to route publications to subscriptions by defining the same clustered topic object on those queue managers. This can be used to improve availability and workload balancing. When you add an extra topic host for the same cluster topic object, you can use the **PUB** parameter to control when publications begin to be routed through the new topic host.

Connecting a queue manager to a publish/subscribe hierarchy
You connect the child queue manager to the parent queue manager in the hierarchy. If the child queue manager is already a member of another hierarchy or cluster, then this connection joins the hierarchies together, or joins the cluster to the hierarchy.

Disconnecting a queue manager from a publish/subscribe hierarchy
Disconnect a child queue manager from a parent queue manager in a publish/subscribe hierarchy.

Defining cluster topics

Creating a single topic space in a publish/subscribe cluster

Scale up a publish/subscribe system to run on multiple queue managers. Use a publish/subscribe cluster to provide each publisher and subscriber with a single identical topic space.

Before you begin

You have implemented a publish/subscribe system on a single version 7 queue manager.

Always create topic spaces with their own root topics, rather than relying on inheriting the attributes of `SYSTEM.BASE.TOPIC`. If you scale your publish/subscribe system up to a cluster, you can define your root topics as cluster topics, on the cluster topic host, and then all your topics are shared throughout the cluster.

About this task

You now want to scale the system up to support more publishers and subscribers and have every topic visible throughout the cluster.

Procedure

1. Create a cluster to use with the publish/subscribe system.

   If you have an existing traditional cluster, for performance reasons it is better to set up a new cluster for the new publish subscribe system. You can use the same servers for the cluster repositories of both clusters
2. Choose one queue manager, possibly one of the repositories, to be the cluster topic host.

3. Ensure every topic that is to be visible throughout the publish/subscribe cluster resolves to an administrative topic object.
   Set the **CLUSTER** attribute naming the publish/subscribe cluster.

What to do next

Connect publisher and subscriber applications to any queue managers in the cluster.

Create administrative topic objects that have the **CLUSTER** attribute. The topics are also propagated throughout the cluster. Publisher and subscriber programs use the administrative topics so that their behavior is not altered by being connected to different queue managers in the cluster

If you need `SYSTEM.BASE.TOPIC` to act like a cluster topic on every queue manager, you need to modify it on every queue manager.

Related concepts

Distributed publish/subscribe networks

Topic spaces

Related tasks

Adding a version 7 or later queue manager to existing Version 6 topic spaces
Extend an existing Version 6 publish/subscribe system to interoperate with a version 7 or later queue manager, sharing the same topic spaces.

Combining the topic spaces of multiple clusters
Create topic spaces that span multiple clusters. Publish to a topic in one cluster and subscribe to it in another.

Combining and isolating topic spaces in multiple clusters
Isolate some topic spaces to a specific cluster, and combine other topic spaces to make them accessible in all the connected clusters.

Publishing and subscribing to topic spaces in multiple clusters
Publish and subscribe to topics in multiple clusters using overlapped clusters. You can use this technique as long as the topic spaces in the clusters do not overlap.

Defining cluster topics

Adding a version 7 or later queue manager to existing Version 6 topic spaces

Extend an existing Version 6 publish/subscribe system to interoperate with a version 7 or later queue manager, sharing the same topic spaces.

Before you begin

You have an existing Version 6 publish/subscribe system.

You have installed IBM MQ version 7 or later on a new server and configured a queue manager.

About this task

You want to extend your existing Version 6 publish/subscribe system to work with version 7 or later queue managers.

You have decided to stabilize development of the Version 6 publish/subscribe system that uses the queued publish/subscribe interface. You intend to add extensions to the system using the version 7 or later MQI. You have no plans now to rewrite the queued publish/subscribe applications.

You intend to upgrade the Version 6 queue managers to version 7 or later in the future. For now, you are continuing to run the existing queued publish/subscribe applications on the version 7 or later queue managers.

Procedure

1. Create one set of sender-receiver channels to connect the version 7 or later queue manager with one of the Version 6 queue managers in both directions.
2. Create two transmission queues with the names of the target queue managers. Use queue manager aliases if you cannot use the name of the target queue manager as the transmission queue name for some reason.
3. Configure the transmission queues to trigger the sender channels.
4. If the Version 6 publish/subscribe system uses streams, add the streams to the version 7 or later queue manager as described in Adding a stream.
5. Check the version 7 or later queue manager **PSMODE** is set to ENABLE.
6. Alter its **PARENT** attribute to refer to one of the Version 6 queue managers.
7. Check the status of the parent-child relationship between the queue managers is active in both directions.

What to do next

Once you have completed the task, both the Version 6 and version 7 or later queue manager share the same topic spaces. For example, you can do all the following tasks.

- Exchange publications and subscriptions between Version 6 and version 7 or later queue managers.
- Run your existing Version 6 publish/subscribe programs on the version 7 or later queue manager.
- View and modify the topic space on either the Version 6 or version 7 or later queue manager.
- Write version 7 or later publish/subscribe applications and run them on the version 7 or later queue manager.
- Create new publications and subscriptions with the version 7 or later applications and exchange them with Version 6 applications.

Related concepts

Distributed publish/subscribe networks

Topic spaces

Related tasks

Creating a single topic space in a publish/subscribe cluster
Scale up a publish/subscribe system to run on multiple queue managers. Use a publish/subscribe cluster to provide each publisher and subscriber with a single identical topic space.

Combining the topic spaces of multiple clusters
Create topic spaces that span multiple clusters. Publish to a topic in one cluster and subscribe to it in another.

Combining and isolating topic spaces in multiple clusters
Isolate some topic spaces to a specific cluster, and combine other topic spaces to make them accessible in all the connected clusters.

Publishing and subscribing to topic spaces in multiple clusters
Publish and subscribe to topics in multiple clusters using overlapped clusters. You can use this technique as long as the topic spaces in the clusters do not overlap.

Defining cluster topics

Combining the topic spaces of multiple clusters

Create topic spaces that span multiple clusters. Publish to a topic in one cluster and subscribe to it in another.

Before you begin

This task assumes that you have existing direct routed publish/subscribe clusters, and you want to propagate some cluster topics into all the clusters.

Note: You cannot do this for topic host routed publish/subscribe clusters.

About this task

To propagate publications from one cluster to another, you need to join the clusters together in a hierarchy; see Figure 66 on page 346. The hierarchical connections propagate subscriptions and publications between the connected queue managers, and the clusters propagate cluster topics within each cluster, but not between clusters.

The combination of these two mechanisms propagates cluster topics between all the clusters. You need to repeat the cluster topic definitions in each cluster.

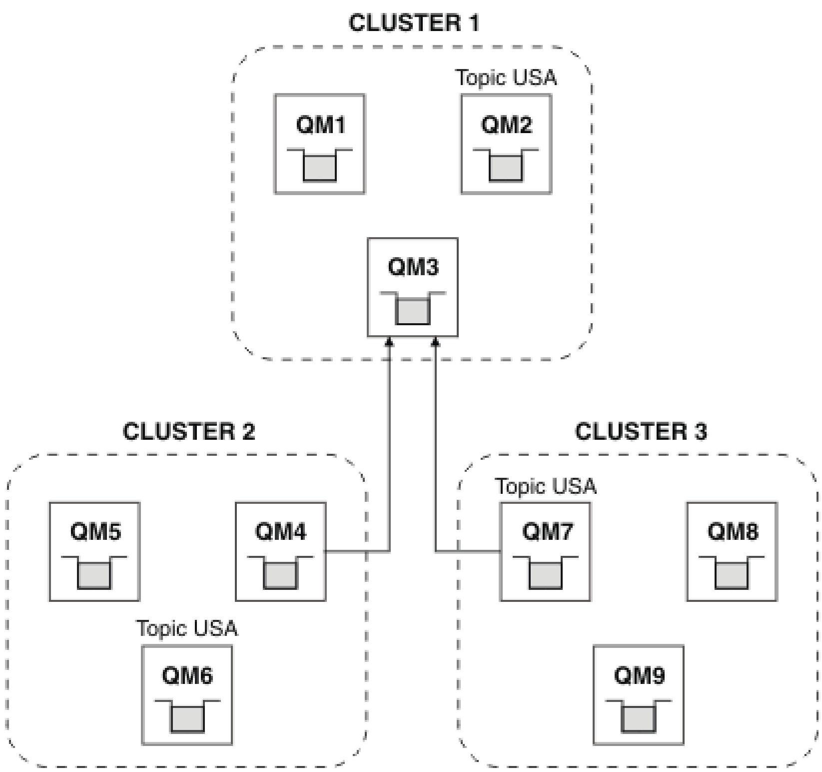

Figure 66. Connecting clusters using hierarchies

The following steps connect the clusters into a hierarchy.

Procedure

1. Create two sets of sender-receiver channels to connect QM3 and QM4, and QM3 and QM7, in both directions. You must use traditional sender-receiver channels and transmission queues, rather than a cluster, to connect a hierarchy.
2. Create three transmission queues with the names of the target queue managers. Use queue manager aliases if you cannot use the name of the target queue manager as the transmission queue name for some reason.
3. Configure the transmission queues to trigger the sender channels.
4. Check the **PSMODE** of QM3, QM4 and QM7 is set to ENABLE.
5. Alter the **PARENT** attribute of QM4 and QM7 to QM3.
6. Check the status of the parent-child relationship between the queue managers is active in both directions.

7. Create the administrative topic USA with the attribute **CLUSTER** ('CLUSTER 1'), **CLUSTER** ('CLUSTER 2'), and **CLUSTER** ('CLUSTER 3') on each of the three cluster topic host queue managers in clusters 1, 2 and 3. The cluster topic host does not need to be a hierarchically connected queue manager.

What to do next

You can now publish or subscribe to the cluster topic USA in Figure 66 on page 346. The publications subscriptions flow to publishers and subscribers in all three clusters.

Suppose that you did not create USA as a cluster topic in the other clusters. If USA is only defined on QM7, then publications and subscriptions to USA are exchanged between QM7, QM8, QM9, and QM3. Publishers and subscribers running on QM7, QM8, QM9 inherit the attributes of the administrative topic USA. Publishers and subscribers on QM3 inherit the attributes of SYSTEM.BASE.TOPIC on QM3.

See also "Combining and isolating topic spaces in multiple clusters" on page 347.

Related concepts

Distributed publish/subscribe networks

Topic spaces

Related tasks

Creating a single topic space in a publish/subscribe cluster
Scale up a publish/subscribe system to run on multiple queue managers. Use a publish/subscribe cluster to provide each publisher and subscriber with a single identical topic space.

Adding a version 7 or later queue manager to existing Version 6 topic spaces
Extend an existing Version 6 publish/subscribe system to interoperate with a version 7 or later queue manager, sharing the same topic spaces.

Combining and isolating topic spaces in multiple clusters
Isolate some topic spaces to a specific cluster, and combine other topic spaces to make them accessible in all the connected clusters.

Publishing and subscribing to topic spaces in multiple clusters
Publish and subscribe to topics in multiple clusters using overlapped clusters. You can use this technique as long as the topic spaces in the clusters do not overlap.

Defining cluster topics

Combining and isolating topic spaces in multiple clusters

Isolate some topic spaces to a specific cluster, and combine other topic spaces to make them accessible in all the connected clusters.

Before you begin

Examine the topic "Combining the topic spaces of multiple clusters" on page 345. It might be sufficient for your needs, without adding an additional queue manager as a bridge.

Note: You can only complete this task using direct routed publish/subscribe clusters. You cannot do this using topic host routed clusters.

About this task

A potential improvement on the topology shown in Figure 66 on page 346 in "Combining the topic spaces of multiple clusters" on page 345 is to isolate cluster topics that are not shared across all the clusters. Isolate clusters by creating a bridging queue manager that is not in any of the clusters; see Figure 67 on page 348. Use the bridging queue manager to filter which publications and subscriptions can flow from one cluster to another.

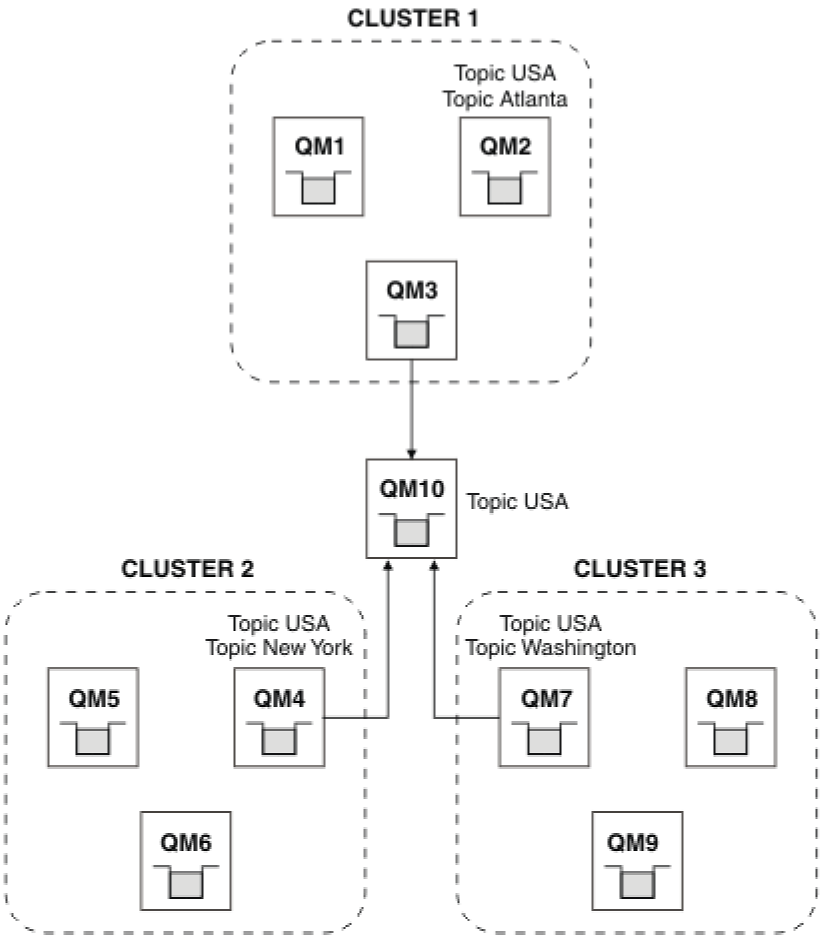

Figure 67. Bridged clusters

Use the bridge to isolate cluster topics that you do not want exposed across the bridge on the other clusters. In Figure 67 on page 348, USA is a cluster topic shared in all the clusters, and `Atlanta`, `New York` and `Washington` are cluster topics that are shared only in one cluster each.

Model your configuration using the following procedure:

Procedure

1. Modify all the `SYSTEM.BASE.TOPIC` topic objects to have **SUBSCOPE** (`QMGR`) and **PUBSCOPE** (`QMGR`) on all the queue managers.

   No topics (even cluster topics) are propagated onto other queue managers unless you explicitly set **SUBSCOPE** (`ALL`) and **PUBSCOPE** (`ALL`) on the root topic of your cluster topics.
2. Define the topics on the three cluster topic host queue managers that you want to be shared in each cluster with the attributes **CLUSTER** (*clustername*), **SUBSCOPE** (`ALL`) and **PUBSCOPE** (`ALL`).

   If you want some cluster topics shared between all the clusters, define the same topic in each of the clusters. Use the cluster name of each cluster as the cluster attribute.
3. For the cluster topics you want shared between all the clusters, define the topics again on the bridge queue manager (`QM10`), with the attributes **SUBSCOPE** (`ALL`), and **PUBSCOPE** (`ALL`).

Example

In the example in Figure 67 on page 348, only topics that inherit from USA propagate between all three clusters.

What to do next

Subscriptions for topics defined on the bridge queue manager with **SUBSCOPE** (ALL) and **PUBSCOPE** (ALL) are propagated between the clusters.

Subscriptions for topics defined within each cluster with attributes **CLUSTER** (*clustername*), **SUBSCOPE** (ALL) and **PUBSCOPE** (ALL) are propagated within each cluster.

Any other subscriptions are local to a queue manager.

Related concepts

Distributed publish/subscribe networks

Topic spaces

Publication scope

Subscription scope

Related tasks

Creating a single topic space in a publish/subscribe cluster
Scale up a publish/subscribe system to run on multiple queue managers. Use a publish/subscribe cluster to provide each publisher and subscriber with a single identical topic space.

Adding a version 7 or later queue manager to existing Version 6 topic spaces
Extend an existing Version 6 publish/subscribe system to interoperate with a version 7 or later queue manager, sharing the same topic spaces.

Combining the topic spaces of multiple clusters
Create topic spaces that span multiple clusters. Publish to a topic in one cluster and subscribe to it in another.

Publishing and subscribing to topic spaces in multiple clusters
Publish and subscribe to topics in multiple clusters using overlapped clusters. You can use this technique as long as the topic spaces in the clusters do not overlap.

Defining cluster topics

Publishing and subscribing to topic spaces in multiple clusters

Publish and subscribe to topics in multiple clusters using overlapped clusters. You can use this technique as long as the topic spaces in the clusters do not overlap.

Before you begin

Create multiple traditional clusters with some queue managers in the intersections between the clusters.

About this task

You might have chosen to overlap clusters for various different reasons.

1. You have a limited number of high availability servers, or queue managers. You decide to deploy all the cluster repositories, and cluster topic hosts to them.
2. You have existing traditional queue manager clusters that are connected using gateway queue managers. You want to deploy publish/subscribe applications to the same cluster topology.
3. You have a several self contained publish/subscribe applications. For performance reasons, it is better to keep publish/subscribe clusters small and separate from traditional clusters. You have decided to deploy the applications to different clusters. However, you also want to monitor all the publish/subscribe applications on one queue manager, as you have licensed only one copy of the monitoring application. This queue manager must have access to the publications to cluster topics in all the clusters.

By ensuring that your topics are defined in non-overlapping topic spaces, you can deploy the topics into overlapping publish/subscribe clusters, see Figure 68 on page 350. If the topic spaces overlap, then deploying to overlapping clusters leads to problems.

Because the publish/subscribe clusters overlap you can publish and subscribe to any of the topic spaces using the queue managers in the overlap.

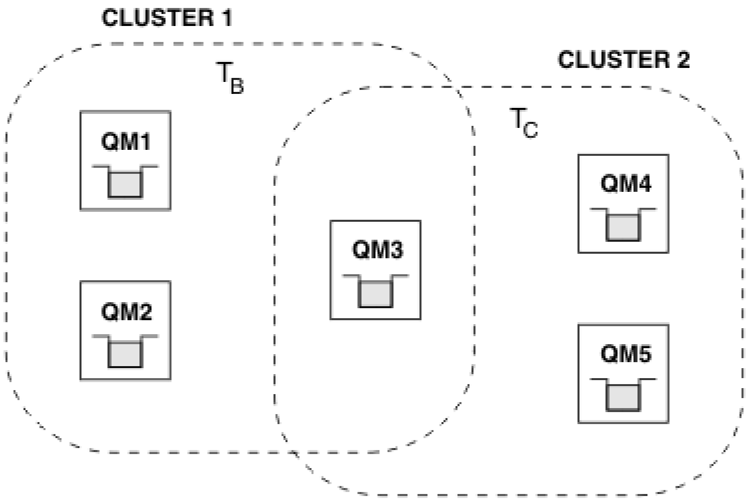

Figure 68. Overlapping clusters, non-overlapping topic spaces

Procedure

Create a means of ensuring that topic spaces do not overlap.

For example, define a unique root topic for each of the topic spaces. Make the root topics cluster topics.

a) `DEFINE TOPIC(B) TOPICSTR('B') CLUSTER('CLUSTER 1') ...`

b) `DEFINE TOPIC(C) TOPICSTR('C') CLUSTER('CLUSTER 2') ...`

Example

In Figure 68 on page 350 publishers and subscriber connected to QM3 can publish or subscribe to T_B or T_C

What to do next

Connect publishers and subscribers that use topics in both clusters to queue managers in the overlap.

Connect publishers and subscribers that must only use topics in a specific cluster to queue managers not in the overlap.

Related concepts

Distributed publish/subscribe networks

Topic spaces

Related tasks

Creating a single topic space in a publish/subscribe cluster
Scale up a publish/subscribe system to run on multiple queue managers. Use a publish/subscribe cluster to provide each publisher and subscriber with a single identical topic space.

Adding a version 7 or later queue manager to existing Version 6 topic spaces
Extend an existing Version 6 publish/subscribe system to interoperate with a version 7 or later queue manager, sharing the same topic spaces.

Combining the topic spaces of multiple clusters
Create topic spaces that span multiple clusters. Publish to a topic in one cluster and subscribe to it in another.

Combining and isolating topic spaces in multiple clusters
Isolate some topic spaces to a specific cluster, and combine other topic spaces to make them accessible in all the connected clusters.

Defining cluster topics

Connecting a queue manager to a publish/subscribe hierarchy

You connect the child queue manager to the parent queue manager in the hierarchy. If the child queue manager is already a member of another hierarchy or cluster, then this connection joins the hierarchies together, or joins the cluster to the hierarchy.

Before you begin

1. Queue managers in a publish/subscribe hierarchy must have unique queue manager names.
2. A publish/subscribe hierarchy relies on the "queued publish/subscribe" queue manager feature. This must be enabled on both the parent and the child queue managers. See Starting queued publish/subscribe.
3. The publish/subscribe relationship relies on queue manager sender and receiver channels. There are two ways to establish the channels:
   - Add both the parent and child queue managers to a IBM MQ cluster. See Adding a queue manager to a cluster.
   - Establish a sender/receiver channel pair from the child queue manager to the parent and from the parent to the child. Each channel either needs to use a transmission queue with the same name as the target queue manager, or a queue manager alias with the same name as the target queue manager. For more information about how to establish a point-to-point channel connection, see IBM MQ distributed queuing techniques.

   For examples that configure a hierarchy over each type of channel configuration, see the following set of publish/subscribe hierarchy scenarios:

   - Scenario 1: Using point-to-point channels with queue manager name alias
   - Scenario 2: Using point-to-point channels with same name for transmission queue and remote queue manager
   - Scenario 3: Using a cluster channel to add a queue manager

About this task

Use the `ALTER QMGR PARENT (`*PARENT_NAME*`)` **runmqsc** command to connect children to parents. This configuration is performed on the child queue manager, where *PARENT_NAME* is the name of the parent queue manager.

Procedure

ALTER QMGR PARENT(*PARENT_NAME*)

Example

The first example shows how to attach queue manager QM2 as a child of QM1, then query QM2 to confirm it has successfully become a child with a **STATUS** of `ACTIVE`:

```
C:>runmqsc QM2
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM2
alter qmgr parent(QM1)
     1 : alter qmgr parent(QM1)
AMQ8005: IBM MQ queue manager changed.
display pubsub all
     2 : display pubsub all
AMQ8723: Display pub/sub status details.
   QMNAME(QM2)                         TYPE(LOCAL)
   STATUS(ACTIVE)
AMQ8723: Display pub/sub status details.
   QMNAME(QM1)                         TYPE(PARENT)
STATUS(ACTIVE)
```

The next example shows the result of querying QM1 for its connections:

```
C:\Documents and Settings\Admin>runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.
display pubsub all
     2 : display pubsub all
AMQ8723: Display pub/sub status details.
   QMNAME(QM1)                          TYPE(LOCAL)
   STATUS(ACTIVE)
AMQ8723: Display pub/sub status details.
   QMNAME(QM2)                          TYPE(CHILD)
   STATUS(ACTIVE)
```

If **STATUS** does not show as ACTIVE, check that the channels between the child and the parent are correctly configured and running. Check both queue manager error logs for possible errors.

What to do next

By default, topics used by publishers and subscribers on one queue manager are shared with publishers and subscribers on the other queue managers in the hierarchy. Administered topics can be configured to control the level of sharing through use of the **SUBSCOPE** and **PUBSCOPE** topic properties. See Configuring distributed publish/subscribe networks.

Related concepts

Combining publication and subscription scopes
In IBM MQ versions 7 onwards, publication and subscription scope work independently to determine the flow of publications between queue managers.

Combining topic spaces in publish/subscribe networks
Combine the topic space of a queue manager with other queue managers in a publish/subscribe cluster or hierarchy. Combine publish/subscribe clusters, and publish/subscribe clusters with hierarchies.

Streams and topics

Publish/subscribe messaging

Related tasks

Configuring a publish/subscribe cluster
Define a topic on a queue manager. To make the topic a cluster topic, set the **CLUSTER** property. To choose the routing to use for publications and subscriptions for this topic, set the **CLROUTE** property.

Moving a cluster topic definition to a different queue manager
For either topic host routed or direct routed clusters, you might need to move a cluster topic definition when decommissioning a queue manager, or because a cluster queue manager has failed or is unavailable for a significant period of time.

Adding extra topic hosts to a topic host routed cluster
In a topic host routed publish/subscribe cluster, multiple queue managers can be used to route publications to subscriptions by defining the same clustered topic object on those queue managers. This can be used to improve availability and workload balancing. When you add an extra topic host for the same cluster topic object, you can use the **PUB** parameter to control when publications begin to be routed through the new topic host.

Disconnecting a queue manager from a publish/subscribe hierarchy
Disconnect a child queue manager from a parent queue manager in a publish/subscribe hierarchy.

Related reference

DISPLAY PUBSUB

Disconnecting a queue manager from a publish/subscribe hierarchy

Disconnect a child queue manager from a parent queue manager in a publish/subscribe hierarchy.

About this task

Use the **ALTER QMGR** command to disconnect a queue manager from a broker hierarchy. You can disconnect a queue manager in any order at any time.

The corresponding request to update the parent is sent when the connection between the queue managers is running.

Procedure

```
ALTER QMGR PARENT( '')
```

Example

```
C:\Documents and Settings\Admin>runmqsc QM2
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM2.
    1 : alter qmgr parent('')
AMQ8005: IBM MQ queue manager changed.
    2 : display pubsub type(child)
AMQ8147: IBM MQ object  not found.
display pubsub type(parent)
    3 : display pubsub type(parent)
AMQ8147: IBM MQ object  not found.
```

What to do next

You can delete any streams, queues and manually defined channels that are no longer needed.

Related concepts

Combining publication and subscription scopes
In IBM MQ versions 7 onwards, publication and subscription scope work independently to determine the flow of publications between queue managers.

Combining topic spaces in publish/subscribe networks
Combine the topic space of a queue manager with other queue managers in a publish/subscribe cluster or hierarchy. Combine publish/subscribe clusters, and publish/subscribe clusters with hierarchies.

Related tasks

Configuring a publish/subscribe cluster
Define a topic on a queue manager. To make the topic a cluster topic, set the **CLUSTER** property. To choose the routing to use for publications and subscriptions for this topic, set the **CLROUTE** property.

Moving a cluster topic definition to a different queue manager
For either topic host routed or direct routed clusters, you might need to move a cluster topic definition when decommissioning a queue manager, or because a cluster queue manager has failed or is unavailable for a significant period of time.

Adding extra topic hosts to a topic host routed cluster
In a topic host routed publish/subscribe cluster, multiple queue managers can be used to route publications to subscriptions by defining the same clustered topic object on those queue managers. This can be used to improve availability and workload balancing. When you add an extra topic host for the same cluster topic object, you can use the **PUB** parameter to control when publications begin to be routed through the new topic host.

Connecting a queue manager to a publish/subscribe hierarchy
You connect the child queue manager to the parent queue manager in the hierarchy. If the child queue manager is already a member of another hierarchy or cluster, then this connection joins the hierarchies together, or joins the cluster to the hierarchy.

Configuring multiple installations

When using multiple installations on the same system, you must configure the installations and queue managers.

This information applies to UNIX, Linux, and Windows.

Use the information in the following links to configure your installations:

- "Changing the primary installation" on page 362

- “Associating a queue manager with an installation” on page 364
- “Connecting applications in a multiple installation environment” on page 354

Related concepts

Multiple installations

Related tasks

Choosing a primary installation

Choosing an installation name

Connecting applications in a multiple installation environment

On UNIX, Linux, and Windows systems, if IBM WebSphere MQ 7.1, or later, libraries are loaded, IBM MQ automatically uses the appropriate libraries without you needing to take any further action. IBM MQ uses libraries from the installation associated with the queue manager that the application connects to.

The following concepts are used to explain the way applications connect to IBM MQ:

Linking
When the application is compiled, the application is linked to the IBM MQ libraries to get the function exports that are then loaded when the application runs.

Loading
When the application is run, the IBM MQ libraries are located and loaded. The specific mechanism used to locate the libraries varies by operating system, and by how the application is built. For more information about how to locate and load libraries in a multiple installation environment, see “Loading IBM WebSphere MQ 7.1, or later version, libraries” on page 356.

Connecting
When the application connects to a running queue manager, for example, using a `MQCONN` or `MQCONNX` call, it connects using the loaded IBM MQ libraries.

When a server application connects to a queue manager, the loaded libraries must come from the installation associated with the queue manager. With multiple installations on a system, this restriction introduces new challenges when choosing the mechanism that the operating system uses to locate the IBM MQ libraries to load:

- When the **`setmqm`** command is used to change the installation associated with a queue manager, the libraries that need to be loaded change.
- When an application connects to multiple queue managers that are owned by different installations, multiple sets of libraries need to be loaded.

However, if IBM WebSphere MQ 7.1, or later, libraries, are located and loaded, IBM MQ then loads and uses the appropriate libraries without you needing to take any further action. When the application connects to a queue manager, IBM MQ loads libraries from the installation that the queue manager is associated with.

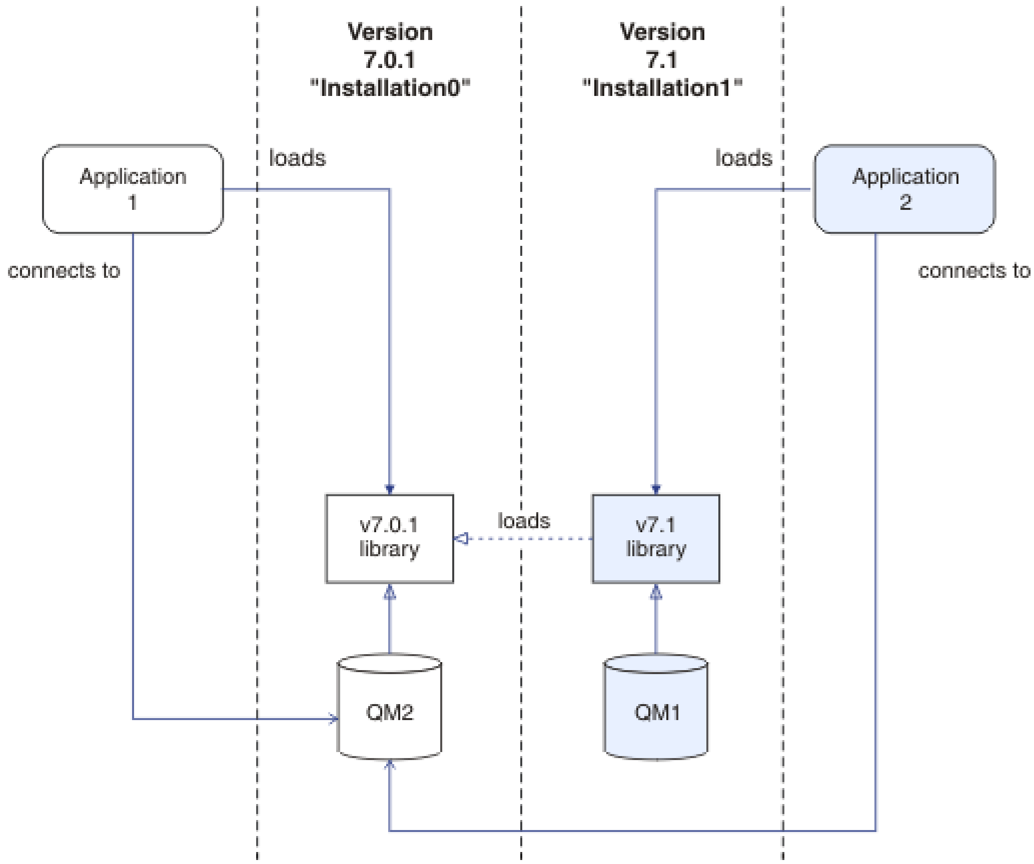

Figure 69. Connecting applications in a multiple installation environment

For example, Figure 69 on page 355 shows a multiple installation environment with a Version 7.0.1 installation (`Installation0`), and a Version 7.1 installation (`Installation1`). Two applications are connected to these installations, but they load different library versions.

`Application 1` directly loads a Version 7.0.1 library. When `application 1` connects to `QM2`, the Version 7.0.1 libraries are used . If `application 1` attempts to connect to `QM1`, or if `QM2` is associated with `Installation1`, `application 1` fails with a 2059 (080B) (RC2059): MQRC_Q_MGR_NOT_AVAILABLE error. The application fails because the Version 7.0.1 library is not capable of loading other library versions. That is, if Version 7.0.1 libraries are directly loaded, you cannot use a queue manager associated with an installation at a later version of IBM MQ.

`Application 2` directly loads a Version 7.1 library. When `application 2` connects to `QM2`, the Version 7.1 library then loads and uses the Version 7.0.1 library. If `application 2` connects to `QM1`, or if `QM2` is associated with `Installation1`, the Version 7.1 library is loaded, and the application works as expected.

Migration scenarios and connecting applications with multiple installations is considered in more detail in Multi-installation queue manager coexistence on UNIX, Linux, and Windows .

For more information about how to load IBM WebSphere MQ 7.1 libraries, see “Loading IBM WebSphere MQ 7.1, or later version, libraries” on page 356.

Support and restrictions

If any of the following Version 7.1, or later, libraries, are located and loaded, IBM MQ can automatically load and use the appropriate libraries:

- The C server libraries
- The C++ server libraries

- The XA server libraries
- The COBOL server libraries
- The COM+ server libraries
- .NET in unmanaged mode

IBM MQ also automatically loads and uses the appropriate libraries for Java and JMS applications in bindings mode.

There are a number of restrictions for applications using multiple installations. For more information, see "Restrictions for applications using multiple installations" on page 359.

Related concepts

"Associating a queue manager with an installation" on page 364
When you create a queue manager, it is automatically associated with the installation that issued the **crtmqm** command. On UNIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

"Restrictions for applications using multiple installations" on page 359
There are restrictions when using CICS server libraries, fast path connections, message handles, and exits in a multiple installation environment.

"Loading IBM WebSphere MQ 7.1, or later version, libraries" on page 356
When deciding how to load IBM MQ libraries, you need to consider a number of factors, including: your environment, whether you can change your existing applications, whether you want a primary installation, where IBM MQ is installed, and whether the location of IBM MQ is likely to change.

Related tasks

Choosing a primary installation

"Changing the primary installation" on page 362
You can use the **setmqinst** command to set or unset an installation as the primary installation.

Loading IBM WebSphere MQ 7.1, or later version, libraries

When deciding how to load IBM MQ libraries, you need to consider a number of factors, including: your environment, whether you can change your existing applications, whether you want a primary installation, where IBM MQ is installed, and whether the location of IBM MQ is likely to change.

How IBM WebSphere MQ 7.1, or later version, libraries are located and loaded depends on your installation environment:

- On UNIX and Linux systems, if a copy of IBM WebSphere MQ 7.1, or later version, is installed in the default location, existing applications continue to work in the same way as previous versions. However, if the applications need symbolic links in `/usr/lib`, you must either select a Version 7.1, or later version, installation to be the primary installation, or manually create the symbolic links.
- If IBM WebSphere MQ 7.1, or later version, is installed in a non-default location, which is the case if IBM WebSphere MQ 7.0.1 is also installed, you might need to change your existing applications so that the correct libraries are loaded.

How IBM WebSphere MQ 7.1, or later version, libraries can be located and loaded also depends on how any existing applications are set up to load libraries. For more information about how libraries can be loaded, see "Operating system library loading mechanisms" on page 358.

Optimally, you should ensure the IBM MQ library, that is loaded by the operating system, is the one with which the queue manager is associated.

The methods for loading IBM MQ libraries vary by platform, and each method has benefits and drawbacks.

Table 27. Benefits and drawbacks of the options for loading libraries			
Platform	**Option**	**Benefits**	**Drawbacks**
UNIX and Linux systems	Set or change the embedded runtime search path (RPath) of the application. This option requires you to recompile and link the application. For more information about compiling and linking applications, see Building a procedural application.	• Scope of the change is clear.	• You must be able to recompile and link the application. • If the location of IBM MQ changes, you must change the RPath.
UNIX and Linux systems	Set the *LD_LIBRARY_PATH* environment variable (*LIBPATH* on AIX), using setmqenv, or crtmqenv, with the **-k** or **-l** option.	• No changes to existing applications required. • Overrides embedded RPaths in an application. • Easy to change the variable if the location of IBM MQ changes.	• `setuid` and `setgid` applications, or applications built in other ways, might ignore *LD_LIBRARY_PATH* for security reasons. • Environment specific, so must be set in each environment where the application is run. • Possible impact on other applications that rely on *LD_LIBRARY_PATH*. • HP-UX: Options used when the application was compiled might disable the use of *LD_LIBRARY_PATH*. For more information, see Runtime linking considerations for HP-UX. • Linux: The compiler used to build the application might disable the use of *LD_LIBRARY_PATH*. For more information, see Runtime linking considerations for Linux .
Windows systems	Set the PATH variable using setmqenv, or crtmqenv.	• No changes required for existing applications. • Easy to change the variable if the location of IBM MQ changes.	• Environment specific, so must be set in each environment where the application is run. • Possible impact on other applications.

Table 27. Benefits and drawbacks of the options for loading libraries (continued)			
Platform	**Option**	**Benefits**	**Drawbacks**
UNIX, Linux, and Windows systems	Set the primary installation to a Version 7.1, or later, installation. See "Changing the primary installation" on page 362. For more information about the primary installation, see Choosing a primary installation.	• No changes required for existing applications. • Easy to change the primary installation if the location of IBM MQ changes. • Gives similar behavior to previous versions of IBM MQ.	• When IBM WebSphere MQ 7.0.1 is installed, you cannot set the primary installation to Version 7.1, or later. • UNIX and Linux: Does not work if `/usr/lib` is not in the default search path.

Library loading considerations for HP-UX

The sample compilation commands in the product documentation for previous versions of IBM MQ included the `-W1, +noenvvar` link option for 64-bit applications. This option disables the use of *LD_LIBRARY_PATH* to load shared libraries. If you want your applications to load IBM MQ libraries from a location other than the location specified in the RPath, you must update your applications. You can update the applications by recompiling and linking without the `-W1, +noenvvar` link option, or by using the **`chatr`** command.

To find out how your applications currently load libraries, see "Operating system library loading mechanisms" on page 358.

Library loading considerations for Linux

Applications compiled using some versions of gcc, for example, version 3.2.x, can have an embedded RPath that cannot be overridden using the *LD_LIBRARY_PATH* environment variable. You can determine if an application is affected by using the `readelf -d` *`applicationName`* command. The RPath cannot be overridden if the RPATH symbol is present and the RUNPATH symbol is not present.

Library loading considerations for Solaris

The sample compilation commands in the product documentation for previous versions of IBM MQ included the `-lmqmcs -lmqmzse` link options. The appropriate versions of these libraries are now loaded automatically by IBM MQ. If IBM MQ is installed in a non-default location, or if there are multiple installations on the system, you must update your applications. You can update the applications by recompiling and linking without the `-lmqmcs -lmqmzse` link options.

Operating system library loading mechanisms

On Windows systems, several directories are searched to find the libraries:

- The directory the application is loaded from.
- The current directory.
- The directories in the *PATH* environment variable, both the global *PATH* variable and the *PATH* variable of the current user.

On UNIX and Linux systems, there are a number of methods that might have been used to locate the libraries to load:

- Using the *LD_LIBRARY_PATH* environment variable (also *LIBPATH* on AIX, and *SHLIB_PATH* on HP-UX). If this variable is set, it defines a set of directories that are searched for the required IBM MQ libraries. If any libraries are found in these directories, they are used in preference of any libraries that might be found using the other methods.

- Using an embedded search path (RPath). The application might contain a set of directories to search for the IBM MQ libraries. If the *LD_LIBRARY_PATH* is not set, or if the required libraries were not found using the variable, the RPath is searched for the libraries. If your existing applications use an RPath, but you cannot recompile and link the application, you must either install IBM WebSphere MQ 7.1 in the default location, or use another method to find the libraries.
- Using the default library path. If the IBM MQ libraries are not found after searching the *LD_LIBRARY_PATH* variable and RPath locations, the default library path is searched. Usually, this path contains `/usr/lib` or `/usr/lib64`. If the libraries are not found after searching the default library path, the application fails to start because of missing dependencies.

You can use operating system mechanisms to find out if your applications have an embedded search path. For example:

- AIX: **dump**
- HP-UX: **chatr**
- Linux: **readelf**
- Solaris: **elfdump**

Related concepts

"Associating a queue manager with an installation" on page 364
When you create a queue manager, it is automatically associated with the installation that issued the **crtmqm** command. On UNIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

"Restrictions for applications using multiple installations" on page 359
There are restrictions when using CICS server libraries, fast path connections, message handles, and exits in a multiple installation environment.

"Connecting applications in a multiple installation environment" on page 354
On UNIX, Linux, and Windows systems, if IBM WebSphere MQ 7.1, or later, libraries are loaded, IBM MQ automatically uses the appropriate libraries without you needing to take any further action. IBM MQ uses libraries from the installation associated with the queue manager that the application connects to.

Related tasks

Choosing a primary installation

"Changing the primary installation" on page 362
You can use the **setmqinst** command to set or unset an installation as the primary installation.

Restrictions for applications using multiple installations

There are restrictions when using CICS server libraries, fast path connections, message handles, and exits in a multiple installation environment.

CICS server libraries

If you are using the CICS server libraries, IBM MQ does not automatically select the correct library level for you. You must compile and link your applications with the appropriate library level for the queue manager to which the application connects. For more information, see Building libraries for use with TXSeries® for Multiplatforms version 5.

Message handles

Message handles that use the special value of `MQHC_UNASSOCIATED_HCONN` are limited to use with the first installation loaded in a process. If the message handle cannot be used by a particular installation, reason code `MQRC_HMSG_NOT_AVAILABLE` is returned.

This restriction affects message properties. You cannot use message handles to get message properties from a queue manager on one installation and put them to a queue manager on a different installation. For more information about message handles, see MQCRTMH - Create message handle.

Exits

In a multiple installation environment, existing exits must be updated for use with IBM WebSphere MQ 7.1, or later, installations. Data conversion exits generated using the **crtmqcvx** command must be regenerated using the updated command.

All exits must be written using the MQIEP structure, cannot use an embedded RPATH to locate the IBM MQ libraries, and cannot link to the IBM MQ libraries. For more information, see Writing exits and installable services on UNIX, Linux and Windows .

Fast path

On a server with multiple installations, applications using a fast path connection to IBM WebSphere MQ 7.1 or later must follow these rules:

1. The queue manager must be associated with the same installation as the one from which the application loaded the IBM MQ run time libraries. The application must not use a fast path connection to a queue manager associated with a different installation. An attempt to make the connection results in an error, and reason code MQRC_INSTALLATION_MISMATCH.
2. Connecting non-fast path to a queue manager associated with the same installation as the one from which the application has loaded the IBM MQ run time libraries prevents the application connecting fast path, unless either of these conditions are true:
   - The application makes its first connection to a queue manager associated with the same installation a fast path connection.
   - The environment variable, AMQ_SINGLE_INSTALLATION is set.
3. Connecting non-fast path to a queue manager associated with a Version 7.1 or later installation, has no effect on whether an application can connect fast path.
4. You cannot combine connecting to a queue manager associated with a Version 7.0.1 installation and connecting fast path to a queue manager associated with a Version 7.1, or later installation.

With AMQ_SINGLE_INSTALLATION set, you can make any connection to a queue manager a fast path connection. Otherwise almost the same restrictions apply:

- The installation must be the same one from which the IBM MQ run time libraries were loaded.
- Every connection on the same process must be to the same installation. If you attempt to connect to a queue manager associated with a different installation, the connection fails with reason code MQRC_INSTALLATION_MISMATCH. Note that with AMQ_SINGLE_INSTALLATION set, this restriction applies to all connections, not only fast path connections.
- Only connect one queue manager with fast path connections.

Related reference

MQCONNX - Connect queue manager (extended)

MQIEP structure

2583 (0A17) (RC2583): MQRC_INSTALLATION_MISMATCH

2587 (0A1B) (RC2587): MQRC_HMSG_NOT_AVAILABLE

2590 (0A1E) (RC2590): MQRC_FASTPATH_NOT_AVAILABLE

Connecting .NET applications in a multiple installation environment

By default, applications use the .NET assemblies from the primary installation. If there is no primary installation, or you do not want to use the primary installation assemblies, you must update the application configuration file, or the *DEVPATH* environment variable.

If there is a primary installation on the system, the .NET assemblies and policy files of that installation are registered to the global assembly cache (GAC). The .NET assemblies for all other installations can be found in the installation path of each installation, but the assemblies are not registered to the GAC. Therefore, by default, applications run using the .NET assemblies from the primary installation. You must update the application configuration file if any of the following cases are true:

- You do not have a primary installation.
- You do not want the application to use the primary installation assemblies.
- The primary installation is a lower version of IBM MQ than the version that the application was compiled with.

For information about how to update the application configuration file, see "Connecting .NET applications using the application configuration file" on page 361.

You must update the *DEVPATH* environment variable if the following case is true:

- You want your application to use the assemblies from a non-primary installation, but the primary installation is at the same version as the non-primary installation.

For more information about how to update the *DEVPATH* variable, see "Connecting .NET applications using DEVPATH" on page 362.

Connecting .NET applications using the application configuration file

Within the application configuration file, you must set various tags to redirect applications to use assemblies that are not from the primary installation.

The following table shows the specific changes that need to be made to the application configuration file to allow .NET applications connect using particular assemblies:

Table 28. Configuring applications to use particular assemblies

	Applications compiled with an earlier version of IBM MQ	**Applications compiled with a later version of IBM MQ**
To run an application with a later version IBM MQ primary installation. (later version assemblies in GAC):	No changes necessary	No changes necessary
To run an application with an earlier version IBM MQ primary installation. (earlier version assemblies in GAC):	No changes necessary	In the application configuration file: • Use the *<bindingRedirect>* tag to indicate the use of the earlier version of the assemblies that are in the GAC
To run an application with a later version of IBM MQ non-primary installation. (later version assemblies in installation folder):	In the application configuration file: • Use the *<codebase>* tag to point to the location of the later version assemblies • Use the *<bindingRedirect>* tag to indicate the use of the later version assemblies	In the application configuration file: • Use the *<codebase>* tag to point to the location of the later version assemblies

Table 28. Configuring applications to use particular assemblies (continued)		
	Applications compiled with an earlier version of IBM MQ	**Applications compiled with a later version of IBM MQ**
To run an application with an earlier version of IBM MQ non-primary installation. (earlier version assemblies in installation folder):	In the application configuration file: • Use the *<codebase>* tag to point to the location of the earlier version assemblies • Include the tag *<publisherpolicy Apply=no>*	In the application configuration file: • Use the *<codebase>* tag to point to the location of the earlier version assemblies • Use the *<bindingRedirect>* tag to indicate the use of the earlier version assemblies • Include the tag *<publisherpolicy Apply=no>*

A sample application configuration file `NonPrimaryRedirect.config` is shipped in the folder *MQ_INSTALLATION_PATH*`\tools\dotnet\samples\base`. This file can be modified with the IBM MQ installation path of any non-primary installation. The file can also be directly included in other configuration files using the *<linkedConfiguration>* tag. Samples are provided for `nmqsget.exe.config` and `nmqsput.exe.config`. Both samples use the *<linkedConfiguration>* tag and include the `NonPrimaryRedirect.config` file.

Connecting .NET applications using DEVPATH

You can find the assemblies using the *DEVPATH* environment variable. The assemblies specified by the *DEVPATH* variable are used in preference to any assemblies in the GAC. See the appropriate Microsoft documentation on *DEVPATH* for more information about when to use this variable.

To find the assemblies using the *DEVPATH* environment variable, you must set the *DEVPATH* variable to the folder that contains the assemblies you want to use. Then, you must then update the application configuration file and add the following runtime configuration information:

```
<configuration>
<runtime>
<developmentMode developerInstallation="true"/>
</runtime>
</configuration>
```

Related concepts

"Connecting applications in a multiple installation environment" on page 354
On UNIX, Linux, and Windows systems, if IBM WebSphere MQ 7.1, or later, libraries are loaded, IBM MQ automatically uses the appropriate libraries without you needing to take any further action. IBM MQ uses libraries from the installation associated with the queue manager that the application connects to.

Multiple installations

Related tasks

Choosing a primary installation

Using .NET

Changing the primary installation

You can use the **`setmqinst`** command to set or unset an installation as the primary installation.

About this task

This task applies to UNIX, Linux, and Windows.

The primary installation is the installation to which required system-wide locations refer. For more information about the primary installation, and considerations for choosing your primary installation, see Choosing a primary installation.

If an installation of IBM WebSphere MQ 7.1 or later is coexisting with an installation of IBM WebSphere MQ 7.0.1, the IBM WebSphere MQ 7.0.1 installation must be the primary. It is flagged as primary when the IBM WebSphere MQ 7.1 or later version is installed, and the IBM WebSphere MQ 7.1 or later installation cannot be made primary.

During the installation process on Windows, you can specify that the installation is to be the primary installation. On UNIX and Linux systems, you must issue a **setmqinst** command after installation to set the installation as the primary installation.

"Set the primary installation" on page 363.

"Unset the primary installation" on page 363.

Set the primary installation

Procedure

To set an installation as the primary installation:

1. Check if an installation is already the primary installation by entering the following command:

   ```
   MQ_INSTALLATION_PATH/bin/dspmqinst
   ```

   where *MQ_INSTALLATION_PATH* is the installation path of a IBM WebSphere MQ 7.1 or later installation.
2. If an existing IBM WebSphere MQ 7.1 or later installation is set as the primary installation, unset it by following the instructions in "Unset the primary installation" on page 363. If IBM WebSphere MQ 7.0.1 is installed on the system, the primary installation cannot be changed.
3. As root on UNIX and Linux systems, or a member of the Administrators group on Windows systems, enter one of the following commands:
   - To set the primary installation using the path of the installation you want to be the primary installation:

     ```
     MQ_INSTALLATION_PATH/bin/setmqinst -i -p MQ_INSTALLATION_PATH
     ```
   - To set the primary installation using the name of the installation you want to be the primary installation:

     ```
     MQ_INSTALLATION_PATH/bin/setmqinst -i -n installationName
     ```
4. On Windows systems, restart the system.

Unset the primary installation

Procedure

To unset an installation as the primary installation:

1. Check which installation is the primary installation by entering the following command:

   ```
   MQ_INSTALLATION_PATH/bin/dspmqinst
   ```

   where *MQ_INSTALLATION_PATH* is the installation path of a IBM WebSphere MQ 7.1 or later installation.

If IBM WebSphere MQ 7.0.1 is the primary installation, you cannot unset the primary installation.

2. As root on UNIX and Linux systems, or a member of the Administrators group on Windows systems, enter one of the following commands:

   - To unset the primary installation using the path of the installation you no longer want to be the primary installation:

     ```
     MQ_INSTALLATION_PATH/bin/setmqinst -x -p MQ_INSTALLATION_PATH
     ```

   - To unset the primary installation using the name of the installation you no longer want to be the primary installation:

     ```
     MQ_INSTALLATION_PATH/bin/setmqinst -x -n installationName
     ```

Related concepts

Features that can be used only with the primary installation on Windows

External library and control command links to primary installation on UNIX and Linux

Related tasks

Uninstalling, upgrading, and maintaining the primary installation

Choosing an installation name

Related reference

setmqinst

Associating a queue manager with an installation

When you create a queue manager, it is automatically associated with the installation that issued the **crtmqm** command. On UNIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

You can use the **setmqm** command in the following ways:

- Moving individual queue managers between equivalent versions of IBM MQ. For example, moving a queue manager from a test to a production system.
- Migrating individual queue managers from an older version of IBM MQ to a newer version of IBM MQ. Migrating queue managers between versions has various implications that you must be aware of. For more information about migrating, see Migrating and upgrading IBM MQ.

To associate a queue manager with an installation:

1. Stop the queue manager using the **endmqm** command from the installation that is currently associated with the queue manager.
2. Associate the queue manager with another installation using the **setmqm** command from that installation.

   For example, to set queue manager QMB to be associated with an installation with the name `Installation2`, enter the following command from Installation2:

   ```
   MQ_INSTALLATION_PATH/bin/setmqm -m QMB -n Installation2
   ```

   where *MQ_INSTALLATION_PATH* is the path where Installation2 is installed.
3. Start the queue manager using the **strmqm** command from the installation that is now associated with the queue manager.

   This command performs any necessary queue manager migration and results in the queue manager being ready to use.

The installation that a queue manager is associated with limits that queue manager so that it can be administered only by commands from that installation. There are three key exceptions:

- **setmqm** changes the installation associated with the queue manager. This command must be issued from the installation that you want to associate with the queue manager, not the installation that the queue manager is currently associated with. The installation name specified by the **setmqm** command has to match the installation from which the command is issued.
- **strmqm** usually has to be issued from the installation that is associated with the queue manager. However, when a V7.0.1 or earlier queue manager is started on a V7.1 or later installation for the first time, **strmqm** can be used. In this case, **strmqm** starts the queue manager and associates it with the installation from which the command is issued.
- **dspmq** displays information about all queue managers on a system, not just those queue managers associated with the same installation as the **dspmq** command. The `dspmq -o installation` command displays information about which queue managers are associated with which installations.

Queue manager association in HA environments

For HA environments, the **addmqinf** command automatically associates the queue manager with the installation from which the **addmqinf** command is issued. As long as the **strmqm** command is then issued from the same installation as the **addmqinf** command, no further setup is required. To start the queue manager using a different installation, you must first change the associated installation using the **setmqm** command.

Queue managers associated with deleted installations

If the installation that a queue manager is associated with has been deleted, or if the queue manager status information is unavailable, the **setmqm** command fails to associate the queue manager with another installation. In this situation, take the following actions:

1. Use the **dspmqinst** command to see the other installations on your system.
2. Manually modify the `InstallationName` field of the `QueueManager` stanza in mqs.ini to specify another installation.
3. Use the **dltmqm** command from that installation to delete the queue manager.

Related concepts

"Finding installations of IBM MQ on a system" on page 365
If you have multiple IBM MQ installations on a system, you can check which versions are installed and where they are.

"The IBM MQ configuration file, mqs.ini" on page 82
The IBM MQ configuration file, mqs.ini, contains information relevant to all the queue managers on the node. It is created automatically during installation.

Related tasks

Choosing a primary installation

Related reference

setmqm

strmqm

dspmq

dspmqinst

Finding installations of IBM MQ on a system

If you have multiple IBM MQ installations on a system, you can check which versions are installed and where they are.

You can use the following methods to find the IBM MQ installations on your system:

- Use the **dspmqver** command. This command does not provide details of all installations on a system if it is issued from a Version 7.0.1 installation.

- Use the platform installation tools to query where IBM MQ has been installed. Then use the **dspmqver** command from a Version 7.1 or later installation. The following commands are examples of commands you can use to query where IBM MQ has been installed:
  - On AIX systems, you can use the **lslpp** command:

    ```
    lslpp -R ALL -l mqm.base.runtime
    ```

  - On HP-UX systems, you can use the **swlist** command:

    ```
    swlist -a location -a revision -l product MQSERIES
    ```

  - On Linux systems, you can use the **rpm** command:

    ```
    rpm -qa --qf "%{NAME}-%{VERSION}-%{RELEASE}\t%{INSTPREFIXES}\n" | grep MQSeriesRuntime
    ```

  - On Solaris systems, you can use the **pkginfo** and **pkgparam** commands:
    1. List the installed packages by entering the following command:

       ```
       pkginfo | grep -w mqm
       ```

    2. For each package listed, enter following command:

       ```
       pkgparam pkgname BASEDIR
       ```

  - On Windows systems, you can use the **wmic** command. This command might install the wmic client:

    ```
    wmic product where "(Name like '%MQ%') AND (not Name like '%bitSupport')" get Name,
    Version, InstallLocation
    ```

- On UNIX and Linux systems, issue the following command to find out where IBM MQ has been installed:

  ```
  cat /etc/opt/mqm/mqinst.ini
  ```

  Then use the **dspmqver** command from a Version 7.1 or later installation.
- To display details of installations on the system, on 32-bit Windows, issue the following command:

  ```
  reg.exe query "HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\Installation" /s
  ```

- On 64-bit Windows, issue the following command:

  ```
  reg.exe query "HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\IBM\WebSphere MQ\Installation" /s
  ```

Note: the **reg.exe** command will only display information for Version 7.1 or later installations.

Related concepts

Multiple installations

Related reference

dspmqver

dspmqinst

Availability, recovery and restart

Make your applications highly available by maintaining queue availability if a queue manager fails, and recover messages after server or storage failure.

Improve client application availability by using client reconnection to switch a client automatically between a group of queue managers, or to the new active instance of a multi-instance queue manager after a queue manager failure. Automatic client reconnect is not supported by IBM MQ classes for Java.

z/OS Improve server application availability on z/OS by using queue sharing groups.

On Windows, IBM i IBM i, UNIX, and Linux platforms deploy server applications to a multi-instance queue manager, which is configured to run as a single queue manager on multiple servers; if the server running the active instance fails, execution is automatically switched to a standby instance of the same queue manager on a different server. If you configure server applications to run as queue manager services, they are restarted when a standby instance becomes the actively running queue manager instance.

You can configure IBM MQ as part of a platform-specific clustering solution such as Microsoft Cluster Server, IBM i HA clusters on IBM i, or PowerHA® for AIX (formerly HACMP on AIX) and other UNIX and Linux clustering solutions.

Another way to increase server application availability is to deploy server applications to multiple computers in a queue manager cluster.

A messaging system ensures that messages entered into the system are delivered to their destination. IBM MQ can trace the route of a message as it moves from one queue manager to another using the **`dspmqrte`** command. If a system fails, messages can be recovered in various ways depending on the type of failure, and the way a system is configured.

IBM MQ ensures that messages are not lost by maintaining recovery logs of the activities of the queue managers that handle the receipt, transmission, and delivery of messages. It uses these logs for three types of recovery:

1. *Restart recovery*, when you stop IBM MQ in a planned way.
2. *Failure recovery*, when a failure stops IBM MQ.
3. *Media recovery*, to restore damaged objects.

In all cases, the recovery restores the queue manager to the state it was in when the queue manager stopped, except that any in-flight transactions are rolled back, removing from the queues any updates that were in-flight at the time the queue manager stopped. Recovery restores all persistent messages; nonpersistent messages might be lost during the process.

Automatic client reconnection

You can make your client applications reconnect automatically, without writing any additional code, by configuring a number of components.

Automatic client reconnection is *inline*. The connection is automatically restored at any point in the client application program, and the handles to open objects are all restored.

In contrast, manual reconnection requires the client application to re-create a connection using `MQCONN` or `MQCONNX`, and to reopen objects. Automatic client reconnection is suitable for many, but not all client applications.

Table 29 on page 368 lists the earliest release of IBM MQ client support that must be installed on a client workstation. You must upgrade client workstations to one of these levels for an application to use automatic client reconnection. Table 30 on page 368 lists other requirements to enable automatic client reconnection.

With program access to reconnection options, a client application can set reconnection options. Except for JMS and XMS clients, if a client application has access to reconnection options, it can also create an event handler to handle reconnection events.

An existing client application might be able to benefit from reconnection support, without recompilation and linking:

- For a non-JMS client, set the `mqclient.ini` environment variable `DefRecon` to set reconnection options. Use a CCDT to connect to a queue manager. If the client is to connect to a multi-instance queue manager, provide the network addresses of the active and standby queue manager instances in the CCDT.
- For a JMS client, set the reconnection options in the connection factory configuration. When running inside the EJB container of a Java EE server, MDBs can reconnect to IBM MQ using the reconnect mechanism provided by activation specifications of the IBM MQ resource adapter (or listener ports if running in WebSphere Application Server). However, if the application is not an MDB (or is running in the web container) the application must implement its own reconnect logic because automatic client reconnect is not supported in this scenario. The IBM MQ resource adapter provides this reconnect ability for the delivery of messages to message driven beans, but other Java EE elements such as servlets must implement their own reconnection.

Note: Automatic client reconnection is not supported by IBM MQ classes for Java.

Table 29. Supported clients

Client interface	Client	Program access to reconnection options	Reconnection support
Messaging APIs	C, C++, COBOL, Unmanaged Visual Basic, XMS (Unmanaged XMS on Windows)	7.0.1	7.0.1
	JMS (JSE, and Java EE client container and managed containers)	7.0.1.3	7.0.1.3
	IBM MQ classes for Java	Not supported	Not supported
	Managed XMS and managed .NET clients: C#, Visual Basic,	7.1	7.1
Other APIs	Windows Communication Foundation (Unmanaged [1])	Not supported	7.0.1
	Windows Communication Foundation (Managed [1])	Not supported	Not supported
	Axis 1	Not supported	Not supported
	Axis 2	Not supported	7.0.1.3
	HTTP (web 2.0)	Not supported	7.0.1.3

1. Set managed or unmanaged mode in the WCF binding configuration.

Automatic reconnection has the following configuration requirements:

Table 30. Automatic reconnection configuration requirements

Component	Requirement	Effect of not meeting requirement
IBM MQ MQI client installation	See Table 29 on page 368	`MQRC_OPTIONS_ERROR`

Table 30. Automatic reconnection configuration requirements (continued)		
Component	**Requirement**	**Effect of not meeting requirement**
IBM MQ Server installation	Level 7.0.1	MQRC_OPTIONS_ERROR
Channel	SHARECNV > 0	MQRC_ENVIRONMENT_ERROR
Application environment	Must be threaded	MQRC_ENVIRONMENT_ERROR
MQI	One of: • MQCONNX with MQCNO Options set to MQCNO_RECONNECT or MQCNO_RECONNECT_Q_MGR. • Defrecon=YES|QMGR in mqclient.ini • In JMS set the CLIENTRECONNECTOPTIONS property of the connection factory.	MQCC_FAILED when a connection is broken or queue manager ends or fails.

Figure 70 on page 369 shows the main interactions between components that are involved in client reconnection.

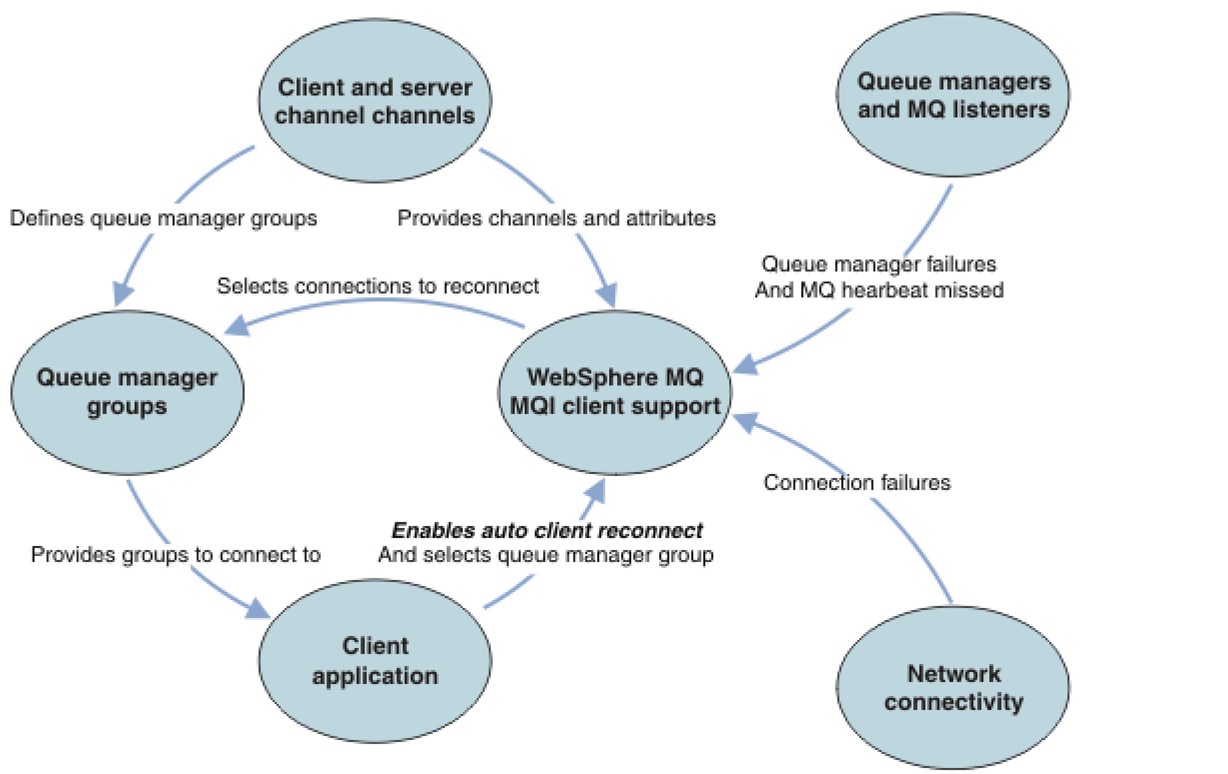

Figure 70. Automatic client reconnection

Client application

The client application is an IBM MQ MQI client.

- By default clients are not automatically reconnected. Enable the automatic client reconnection by setting the MQCONNX MQCNO Option MQCNO_RECONNECT or MQCNO_RECONNECT_Q_MGR.
- Many applications are written in such a way that they are able to take advantage of auto-reconnection with no additional coding. Enable automatic reconnection for existing programs, without making any coding changes, by setting the DefRecon attribute in the channels stanza of the mqclient.ini configuration file.
- Use one of these three options:

  1. Modify the program so that the logic is unaffected by reconnection. For example, you might have to issue MQI calls within the sync point, and resubmit backed-out transactions.
  2. Add an event handler to detect reconnection, and restore the state of the client application when the connection is reestablished.
  3. Do not enable auto-reconnection: instead, disconnect the client and issue a new MQCONN or MQCONNX MQI call to find another queue manager instance that is running in the same queue manager group.

  For further details about these three options, see "Application recovery" on page 450.
- Reconnecting to a queue manager of the same name does not guarantee that you have reconnected to the same instance of a queue manager.

  Use an MQCNO option MQCNO_RECONNECT_Q_MGR, to reconnect to an instance of the same queue manager.
- A client can register an event handler so that it can be informed the state of reconnection. The MQHCONN passed in the event handler cannot be used. The following reason codes are provided:

  MQRC_RECONNECTING
  The connection failed, and the system is attempting to reconnect. You receive multiple MQRC_RECONNECTING events if multiple reconnect attempts are made.

  MQRC_RECONNECTED
  The reconnection made and all handles successfully reestablished.

  MQRC_RECONNECT_FAILED
  The reconnection was not successful.

  MQRC_RECONNECT_QMID_MISMATCH
  A reconnectable connection specified MQCNO_RECONNECT_Q_MGR and the connection attempted to reconnect to a different queue manager.

  MQRC_RECONNECT_Q_MGR_REQD
  An option, such MQMO_MATCH_MSG_TOKEN in an MQGET call, was specified in the client program that requires reconnection to the same queue manager.
- A reconnectable client is able to reconnect automatically only *after* connecting. That is, the MQCONNX call itself is not tried again if it fails. For example, if you receive the return code 2543 - MQRC_STANDBY_Q_MGR from MQCONNX, reissue the call after a short delay.

  MQRC_RECONNECT_INCOMPATIBLE
  This reason code is returned when the application tries to use MQPMO_LOGICAL_ORDER (with MQPUT and MQPUT1) or MQGMO_LOGICAL_ORDER (with MQGET) when reconnect options are set. The reason for returning the reason code is to make sure that applications never use reconnect in such cases.

  MQRC_CALL_INTERRUPTED
  This reason code is returned when the connection breaks during the execution of Commit call and the client reconnects. An MQPUT of a persistent message outside the sync point also results in the same reason code being returned to the application.

Multi-instance queue managers

Simplify restarting IBM MQ MQI client applications, after a multi-instance queue manager has activated its standby instance, by using automatic client reconnection.

The standby instance of a multi-instance queue manager is typically at a different network address to the active instance. Include the network addresses of both the instances in the client connection definition table (CCDT). Either provide a list of network addresses for the **CONNAME** parameter, or define multiple rows for the queue manager in the CCDT.

Commonly, IBM MQ MQI clients reconnect to any queue manager in a queue manager group. Sometimes you want an IBM MQ MQI client to reconnect only to the same queue manager. It might have an affinity to a queue manager. You can prevent a client from reconnecting to a different queue manager. Set the `MQCNO` option, `MQCNO_RECONNECT_Q_MGR`. The IBM MQ MQI client fails if it reconnects to a different queue manager. If you set the `MQCNO` option, `MQCNO_RECONNECT_Q_MGR`, do not include other queue managers in the same queue manager group. The client returns an error if the queue manager it reconnects to is not the same queue manager as the one it connected to.

Queue manager groups

You can select whether the client application always connects and reconnects to a queue manager of the same name, to the same queue manager, or to any of a set of queue managers that are defined with the same `QMNAME` value in the client connection table.

- The queue manager *name* attribute, `QMNAME`, in the client channel definition is the name of a queue manager group.
- In your client application, if you set the value of the `MQCONN` or `MQCONNX` `QmgrName` parameter to a queue manager name, the client connects only to queue managers with that name. If you prefix the queue manager name with an asterisk(*), the client connects to any queue manager in the queue manager group with the same `QMNAME` value. For a full explanation, see Queue manager groups in the CCDT.

Queue sharing groups

Automatic client reconnection to z/OS queue sharing groups, uses the same mechanisms for reconnection as any other environment. The client will reconnect to the same selection of queue managers as is configured for the original connection. For example, when using the client channel definition table the administrator should ensure that all entries in the table, resolve to the same z/OS queue sharing group.

Client and server channel definitions

Client and server channel definitions define the groups of queue managers a client application can reconnect to. The definitions govern the selection and timing of reconnections, and other factors, such as security; see the related topics. The most relevant channel attributes to consider for reconnection are listed in two groups:

Client connection attributes

Connection affinity (AFFINITY) `AFFINITY`
: Connection affinity.

Client channel weight (CLNTWGHT) `CLNTWGHT`
: Client channel weight.

Connection name (CONNAME) `CONNAME`
: Connection information.

Heartbeat interval (HBINT) `HBINT`
: Heartbeat interval. Set the heartbeat interval on the server connection channel.

Keepalive Interval (KAINT) `KAINT`
: Keepalive interval. Set the keepalive interval on the server connection channel.

: Note that KAINT applies to z/OS only.

Queue manager name (QMNAME) `QMNAME`
: Queue manager name.

Server connection attributes

Heartbeat interval (HBINT) HBINT
Heartbeat interval. Set the heartbeat interval on the client connection channel.

Keepalive Interval (KAINT) KAINT
Keepalive interval. Set the keepalive interval on the client connection channel.

Note that KAINT applies to z/OS only.

KAINT is a network layer heartbeat, and HBINT is an IBM MQ heartbeat between the client and the queue manager. Setting these heartbeats to a shorter time serves two purposes:

1. By simulating activity on the connection, network layer software that is responsible for closing inactive connections is less likely to shut down your connection.
2. If the connection is shut down, the delay before the broken connection is detected, is shortened.

The default TCP/IP keepalive interval is two hours. Consider setting the KAINT and HBINT attributes to a shorter time. Do not assume that the normal behavior of a network suits the needs of automatic reconnection. For example, some firewalls can shut down an inactive TCP/IP connection after as little as 10 minutes.

Network connectivity

Only network failures that are passed to the IBM MQ MQI client by the network, are handled by the automatic reconnection capability of the client.

- Reconnections performed automatically by the transport are invisible to IBM MQ.
- Setting HBINT helps to deal with network failures that are invisible to IBM MQ.

Queue managers and IBM MQ listeners

Client reconnection is triggered by server failure, queue manager failure, network connectivity failure, and by an administrator switching over to another queue manager instance.

- If you are using a multi-instance queue manager, an additional cause of client reconnection occurs when you switch control from the active queue manager instance to a standby instance.
- Ending a queue manager using the default **endmqm** command, does not trigger automatic client reconnection. Add the `-r` option on the **endmqm** command to request automatic client reconnection, or the `-s` option to transfer to a standby queue manager instance after shutting down.

IBM MQ MQI client automatic reconnection support

If you use the automatic client reconnection support in the IBM MQ MQI client, the client application automatically reconnects and continues processing without you issuing an MQCONN or MQCONNX MQI call to reconnect to the queue manager.

- Automatic client reconnection is triggered by one of the following occurrences:
  - queue manager failure
  - ending a queue manager and specifying the `-r`, reconnect, option on the **endmqm** command
- The MQCONNX MQCNO options control whether you have enabled the automatic client reconnection. The options are described in Reconnection options.
- Automatic client reconnection issues MQI calls on behalf of your application to restore the connection handle and the handles to other open objects, so that your program can resume normal processing after it has processed any MQI errors that resulted from the broken connection. See "Recovery of an automatically reconnected client" on page 452.
- If you have written a channel exit program for the connection, the exit receives these additional MQI calls.
- You can register a reconnection event handler, which is triggered when reconnection begins and when it finishes.

Although the intended reconnection time is no more than a minute, reconnection can take longer because a queue manager might have numerous resources to manage. During this time, a client application might be holding locks that do not belong to IBM MQ resources. There is a timeout value you can configure to limit the time a client waits for reconnection. The value (in seconds) is set in the `mqclient.ini` file.

```
Channels:
MQReconnectTimeout = 1800
```

No reconnection attempts are made after the timeout has expired. When the system detects that the timeout has expired it returns a `MQRC_RECONNECT_FAILED` error.

z/OS Console message monitoring

There are a number of information messages issued by the queue manager or channel initiator that should be considered particularly significant. These messages do not in themselves indicate a problem, but may be useful in tracking because they do indicate a potential issue which might need addressing.

The presence of these console messages might also indicate that a user application is putting a large number of messages to the page set, which might be a symptom of a larger problem:

- A problem with the user application which PUTs messages, such as an uncontrolled loop.
- A user application which GETs the messages from the queue is no longer functioning.

Cluster error recovery for servers on distributed platforms

distributed

From IBM WebSphere MQ 7.1 onwards, the queue manager reruns operations that caused problems, until the problems are resolved. See Changes to cluster error recovery on servers other than z/OS for more information.

Console messages to monitor

z/OS

The following list outlines messages which can potentially indicate larger problems. Determine if it is necessary to track these messages with system automation and provide appropriate documentation so any potential problems can be followed up effectively.

CSQI004I: *csect-name* CONSIDER INDEXING *queue-name* BY *index-type* FOR *connection-type* CONNECTION *connection-name, num-msgs* MESSAGES SKIPPED

- The queue manager has detected an application receiving messages by message ID or correlation ID from a queue that does not have an index defined.
- Consider establishing an index for the identified queue by altering the local queue object, *queue-name*, INDXTYPE attribute to have value *index-type*.

CSQI031I: *csect-name* THE NEW EXTENT OF PAGE SET *psid* HAS FORMATTED SUCCESSFULLY

- Check the curdepth of the queues allocated to this page set.
- Investigate the cause of the failure to process the messages.

.

CSQI041I: *csect-name* JOB *jobname* USER *userid* HAD ERROR ACCESSING PAGE SET *psid*

- Determine if the page set is allocated to the queue manager.
- Issue a **DISPLAY USAGE** command to determine the state of the page set.
- Check the queue manager joblog for additional error messages.

.

CSQI045I: *csect-name* Log RBA has reached *rba*. Plan a log reset

- Plan to stop the queue manager at a convenient time and reset the logs.
- If your queue manager is using 6-byte log RBAs, consider converting the queue manager to use 8-byte log RBAs.

CSQI046E: *csect-name* Log RBA has reached *rba*. Perform a log reset

- Plan to stop the queue manager at a convenient time and reset the logs.
- If your queue manager is using 6-byte log RBAs, consider converting the queue manager to use 8-byte log RBAs.

CSQI047E: *csect-name* Log RBA has reached *rba*. Stop queue manager and reset logs

- Stop the queue manager immediately and reset the logs.
- If your queue manager is using 6-byte log RBAs, consider converting the queue manager to use 8-byte log RBAs.

CSQJ004I: ACTIVE LOG COPY *n* INACTIVE, LOG IN SINGLE MODE, ENDRBA= *ttt*

- The queue manager has activated 'single' logging mode. This is often indicative of a log offload problem.
- Issue a **`DISPLAY LOG`** command to determine your settings for duplexing of active and archive logs. This display also shows how many active logs need offload processing.
- Check the queue manager joblog for additional error messages

CSQJ031D: *csect-name*, THE LOG RBA RANGE MUST BE RESET. REPLY 'Y' TO CONTINUE STARTUP OR 'N' TO SHUTDOWN

- Stop the queue manager and reset the logs as soon as possible and reset the logs.
- If your queue manager is using 6-byte log RBAs, consider converting the queue manager to use 8-byte log RBAs.

CSQJ032E: *csect-name alert-lvl* - APPROACHING END OF THE LOG RBA RANGE OF *max-rba*. CURRENT LOG RBA IS *current-rba*.

- Plan to stop the queue manager and reset the logs as soon as possible.
- If your queue manager is using 6-byte log RBAs, consider converting the queue manager to use 8-byte log RBAs.

CSQJ110E: LAST COPY*n* ACTIVE LOG DATA SET IS *nnn* PERCENT FULL

- Take steps to complete other waiting offload tasks by performing a display request to determine the outstanding requests related to the log offload process. Take the necessary action to satisfy any requests, and permit offload to continue.
- Consider whether there are sufficient active log data sets. If necessary, you can add additional log data sets dynamically by using the DEFINE LOG command.

CSQJ111A: OUT OF SPACE IN ACTIVE LOG DATA SETS

- Perform a display request to ensure that there are no outstanding requests that are related to the log offload process. Take the necessary action to satisfy any requests, and permit offload to continue.
- Consider whether there are sufficient active log data sets. If necessary, you can add additional log data sets dynamically by using the DEFINE LOG command.
- If the delay was caused by the lack of a resource required for offload, the necessary resource must be made available to allow offload to complete and thus permit logging to proceed. For information about recovery from this condition, see Archive log problems.

CSQJ114I: ERROR ON ARCHIVE DATA SET, OFFLOAD CONTINUING WITH ONLY ONE ARCHIVE DATA SET BEING GENERATED

- Check the queue manager joblog for additional error messages.

- Make a second copy of the archive log and update your BSDS manually.

CSQJ115E: OFFLOAD FAILED, COULD NOT ALLOCATE AN ARCHIVE DATA SET
Review the error status information of message CSQJ103E or CSQJ073E. Correct the condition that caused the data set allocation error so that, on retry, the offload can take place.

CSQJ136I: UNABLE TO ALLOCATE TAPE UNIT FOR CONNECTION-ID= *xxxx* CORRELATION-ID= *yyyyyy*, *m* ALLOCATED *n* ALLOWED

- Check the queue manager joblog for additional error messages.

.

CSQJ151I: *csect-name* ERROR READING RBA *rrr*, CONNECTION-ID= *xxxx* CORRELATION-ID= *yyyyyy* REASON CODE= *ccc*

- Check the queue manager joblog for additional messages.
- Issue a **DISPLAY CONN** command to determine which connection is not committing its activity.
- Ensure the application can commit its updates.

CSQJ160I: LONG-RUNNING UOW FOUND, URID= *urid* CONNECTION NAME= *name*

- Check the queue manager joblog for additional messages.
- Issue a **DISPLAY CONN** command to determine which connection is not committing its activity.
- Ensure the application can commit its updates.

CSQJ161I: UOW UNRESOLVED AFTER *n* OFFLOADS, URID= *urid* CONNECTION NAME= *name*

- Determine if the page set is allocated to the queue manager.
- Issue a **DISPLAY USAGE** command to determine the state of the page set.
- Check the queue manager joblog for additional messages.

CSQP011E: CONNECT ERROR STATUS *ret-code* FOR PAGE SET *psid*

- Check the curdepth of the queues allocated to this page set.
- Investigate the cause of the failure to process messages.

CSQP013I: *csect-name* NEW EXTENT CREATED FOR PAGE SET psid. NEW EXTENT WILL NOW BE FORMATTED

- Check the curdepth of the queues allocated to this page set.
- Investigate the cause of failure to process messages.
- Determine if queues need to be relocated to another page set.
- If the volume is full, determine if you need to make the page set a multi volume dataset. If the page set is already multi-volume, consider adding more volumes to the storage group being used. Once more space is available retry the expansion by setting the page set **EXPAND** method to **SYSTEM**. If a retry is required, toggle **EXPAND** to **SYSTEM** and then back to your normal setting.

CSQP014E: *csect-name* EXPANSION FAILED FOR PAGE SET psid. FUTURE REQUESTS TO EXTEND IT WILL BE REJECTED

- Check the curdepth of the queues allocated to this page set.
- Investigate the cause of failure to process messages.
- Determine if queues need to be relocated to another page set.

CSQP016E: *csect-name* PAGE SET *psid* HAS REACHED THE MAXIMUM NUMBER OF EXTENTS. IT CANNOT BE EXTENDED AGAIN

- Check the curdepth of the queues allocated to this page set.
- Investigate the cause of failure to process messages.

CSQP017I: *csect-name* EXPANSION STARTED FOR PAGE SET *psid*
Issue DISPLAY THREAD commands to determine the state of the Units of Work in IBM MQ.

CSQP047E: Unavailable page sets can cause problems - take action to correct this situation

- Follow the System Programmer Response.

CSQQ008I: *nn* units of recovery are still in doubt in queue manager *qqqq*

- Investigate the state of your dead letter queue. Ensure the dead letter queue is not PUT disabled.
- Ensure the dead letter queue is not at the MAXMSG limit.

CSQQ113I: *psb-name region-id* This message cannot be processed

- Check the CSQOUTX dataset to determine the cause of the CSQINPX failure.
- Some commands may not be processed.

CSQX035I: *csect-name* Connection to queue manager *qmgr-name* stopping or broken, MQCC= *mqcc* MQRC= *mqrc* (*mqrc-text*

- Check the MQRC to determine the cause of the failure.
- These codes are documented in IBM MQ for z/OS messages, completion, and reason codes.

CSQX032I: *csect-name* Initialization command handler terminated

- Check the MQRC to determine the cause of the failure.
- These codes are documented in IBM MQ for z/OS messages, completion, and reason codes.

CSQX048I: *csect-name* Unable to convert message for *name*, MQCC= *mqcc* MQRC= *mqrc* (*mqrc-text*)

- Check the joblog to determine the cause of the TCP/IP failure.
- Check the TCP/IP address space for errors.

CSQX234I: *csect-name* Listener stopped, TRPTYPE= *trptype* INDISP= *disposition*

- If the listener does not stop, following a **STOP** command, check the TCP/IP address space for errors.
- Follow the Systems Programmer Response.

CSQX407I: *csect-name* Cluster queue *q-name* definitions inconsistent

- Multiple cluster queues within the cluster have inconsistent values. Investigate and resolve the differences.

CSQX411I: *csect-name* Repository manager stopped

- If the repository manager has stopped because of an error, check the joblog for messages.

CSQX417I: *csect-name* Cluster-senders remain for removed queue manager *qmgr-name*

- Follow the System Programmer Response.

CSQX418I: *csect-name* Only one repository for cluster *cluster-name*

- For increased high availability, clusters should be configured with two full repositories.

CSQX419I: *csect-name* No cluster-receivers for cluster *cluster-name*

- Follow the System Programmer Response.

CSQX420I: *csect-name* No repositories for cluster *cluster-name*

- Follow the System Programmer Response.

CSQX448E: *csect-name* Repository manager stopping because of errors. Restart in *n* seconds

- Follow the System Programmer Response.

This message is put out every 600 seconds (10 minutes) until the SYSTEM.CLUSTER.COMMAND.QUEUE is enabled, by using the command:

```
ALTER QLOCAL(SYSTEM.CLUSTER.COMMAND.QUEUE) GET(ENABLED)
```

Before enabling the queue, manual intervention might be required to resolve the problem that caused the repository manager to end, prior to the first CSQX448E message being issued.

Using IBM MQ with high availability configurations

If you want to operate your IBM MQ queue managers in a high availability (HA) configuration, you can set up your queue managers to work either with a high availability manager, such as PowerHA for AIX (formerly HACMP) or the Microsoft Cluster Service (MSCS), or with IBM MQ multi-instance queue managers.

You need to be aware of the following configuration definitions:

Queue manager clusters
Groups of two or more queue managers on one or more computers, providing automatic interconnection, and allowing queues to be shared among them for load balancing and redundancy.

HA clusters
HA clusters are groups of two or more computers and resources such as disks and networks, connected together and configured in such a way that, if one fails, a high availability manager, such as HACMP (UNIX) or MSCS (Windows) performs a *failover*. The failover transfers the state data of applications from the failing computer to another computer in the cluster and re-initiates their operation there. This provides high availability of services running within the HA cluster. The relationship between IBM MQ clusters and HA clusters is described in "Relationship of HA clusters to queue manager clusters" on page 378.

Multi-instance queue managers
Instances of the same queue manager configured on two or more computers. By starting multiple instances, one instance becomes the active instance and the other instances become standbys. If the active instance fails, a standby instance running on a different computer automatically takes over. You can use multi-instance queue managers to configure your own highly available messaging systems based on IBM MQ, without requiring a cluster technology such as HACMP or MSCS. HA clusters and multi-instance queue managers are alternative ways of making queue managers highly available. Do not combine them by putting a multi-instance queue manager in an HA cluster.

Differences between multi-instance queue managers and HA clusters

Multi-instance queue managers and HA clusters are alternative ways to achieve high availability for your queue managers. Here are some points that highlight the differences between the two approaches.

Multi-instance queue managers include the following features:

- Basic failover support integrated into IBM MQ
- Faster failover than HA cluster
- Simple configuration and operation
- Integration with MQ Explorer

Limitations of multi-instance queue managers include:

- Highly available, high performance networked storage required
- More complex network configuration because queue manager changes IP address when it fails over

HA clusters include the following features:

- The ability to coordinate multiple resources, such as an application server or database
- More flexible configuration options including clusters comprising more than two nodes
- Can failover multiple times without operator intervention
- Takeover of queue manager's IP address as part of the failover

Limitations of HA clusters include:

- Additional product purchase and skills are required

- Disks which can be switched between the nodes of the cluster are required
- Configuration of HA clusters is relatively complex
- Failover is rather slow historically, but recent HA cluster products are improving this
- Unnecessary failovers can occur if there are shortcomings in the scripts that are used to monitor resources such as queue managers

Relationship of HA clusters to queue manager clusters

Queue manager clusters provide load balancing of messages across available instances of queue manager cluster queues. This offers higher availability than a single queue manager because, following a failure of a queue manager, messaging applications can still send messages to, and access, surviving instances of a queue manager cluster queue. However, although queue manager clusters automatically route new messages to the available queue managers in a cluster, messages currently queued on an unavailable queue manager are not available until that queue manager is restarted. For this reason, queue manager clusters alone do not provide high availability of all message data or provide automatic detection of queue manager failure and automatic triggering of queue manager restart or failover. High Availability (HA) clusters provide these features. The two types of cluster can be used together to good effect. For an introduction to queue manager clusters, see Designing clusters.

Using IBM MQ with a high availability cluster on UNIX and Linux

You can use IBM MQ with a high availability (HA) cluster on UNIX and Linux platforms: for example, PowerHA for AIX (formerly HACMP), Veritas Cluster Server, HP Serviceguard, or a Red Hat Enterprise Linux cluster with Red Hat Cluster Suite.

Before IBM WebSphere MQ 7.0.1, SupportPac MC91 was provided to assist in configuring HA clusters. IBM WebSphere MQ 7.0.1 provided a greater degree of control than previous versions over where queue managers store their data. This makes it easier to configure queue managers in an HA cluster. Most of the scripts provided with SupportPac MC91 are no longer required, and the SupportPac is withdrawn.

This section introduces “HA cluster configurations” on page 378, the relationship of HA clusters to queue manager clusters, “IBM MQ clients” on page 379, and “IBM MQ operating in an HA cluster” on page 379, and guides you through the steps and provides example scripts that you can adapt to configure queue managers with an HA cluster.

Refer to the HA cluster documentation particular to your environment for assistance with the configuration steps described in this section.

HA cluster configurations

In this section the term *node* is used to refer to the entity that is running an operating system and the HA software; "computer", "system" or "machine" or "partition" or "blade" might be considered synonyms in this usage. You can use IBM MQ to help set up either standby or takeover configurations, including mutual takeover where all cluster nodes are running IBM MQ workload.

A *standby* configuration is the most basic HA cluster configuration in which one node performs work while the other node acts only as standby. The standby node does not perform work and is referred to as idle; this configuration is sometimes called *cold standby*. Such a configuration requires a high degree of hardware redundancy. To economize on hardware, it is possible to extend this configuration to have multiple worker nodes with a single standby node. The point of this is that the standby node can take over the work of any other worker node. This configuration is still referred to as a standby configuration and sometimes as an "N+1" configuration.

A *takeover* configuration is a more advanced configuration in which all nodes perform some work and critical work can be taken over in the event of a node failure.

A *one-sided takeover* configuration is one in which a standby node performs some additional, noncritical and unmovable work. This configuration is similar to a standby configuration but with (noncritical) work being performed by the standby node.

A *mutual takeover* configuration is one in which all nodes are performing highly available (movable) work. This type of HA cluster configuration is also sometimes referred to as "Active/Active" to indicate that all nodes are actively processing critical workload.

With the extended standby configuration or either of the takeover configurations it is important to consider the peak load that might be placed on a node that can take over the work of other nodes. Such a node must possess sufficient capacity to maintain an acceptable level of performance.

Relationship of HA clusters to queue manager clusters

Queue manager clusters reduce administration and provide load balancing of messages across instances of queue manager cluster queues. They also offer higher availability than a single queue manager because, following a failure of a queue manager, messaging applications can still access surviving instances of a queue manager cluster queue. However, queue manager clusters alone do not provide automatic detection of queue manager failure and automatic triggering of queue manager restart or failover. HA clusters provide these features. The two types of cluster can be used together to good effect.

IBM MQ clients

IBM MQ clients that are communicating with a queue manager that might be subject to a restart or takeover must be written to tolerate a broken connection and must repeatedly attempt to reconnect. IBM WebSphere MQ Version 7 introduced features in the processing of the Client Channel Definition Table (CCDT) that assist with connection availability and workload balancing; however these are not directly relevant when working with a failover system.

Transactional functionality allows an IBM MQ MQI client to participate in two-phase transactions, as long as the client is connected to the same queue manager. Transactional functionality cannot use techniques, such as an IP load balancer, to select from a list of queue managers. When you use an HA product, a queue manager maintains its identity (name and address) whichever node it is running on, so transactional functionality can be used with queue managers that are under HA control.

IBM MQ operating in an HA cluster

All HA clusters have the concept of a unit of failover. This is a set of definitions that contains all the resources that make up the highly available service. The unit of failover includes the service itself and all other resources upon which it depends.

HA solutions use different terms for a unit of failover:

- On PowerHA for AIX the unit of failover is called a *resource group*.
- On Veritas Cluster Server it is known as a *service group*.
- On Serviceguard it is called a *package*.

This topic uses the term *resource group* to mean a unit of failover.

The smallest unit of failover for IBM MQ is a queue manager. Typically, the resource group containing the queue manager also contains shared disks in a volume group or disk group that is reserved exclusively for use by the resource group, and the IP address that is used to connect to the queue manager. It is also possible to include other IBM MQ resources, such as a listener or a trigger monitor in the same resource group, either as separate resources, or under the control of the queue manager itself.

A queue manager that is to be used in an HA cluster must have its data and logs on disks that are shared between the nodes in the cluster. The HA cluster ensures that only one node in the cluster at a time can write to the disks. The HA cluster can use a monitor script to monitor the state of the queue manager.

It is possible to use a single shared disk for both the data and logs that are related to the queue manager. However, it is normal practice to use separate shared file systems so that they can be independently sized and tuned.

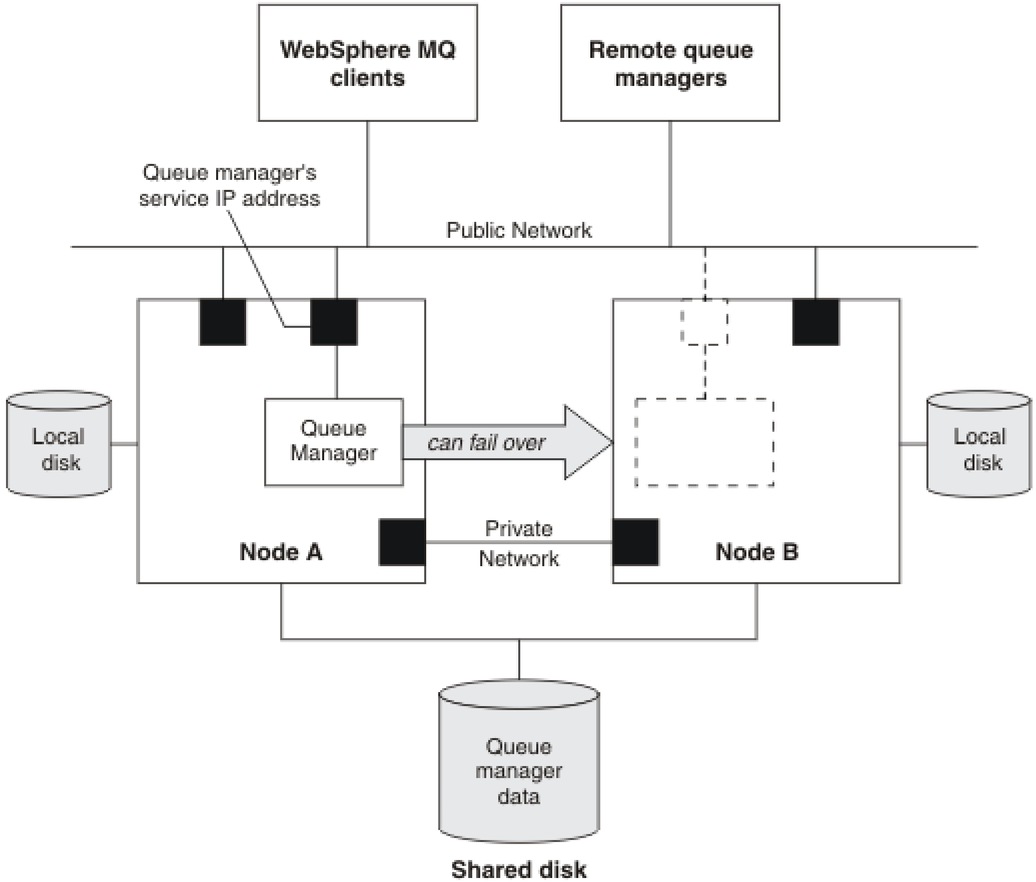

Figure 71. HA cluster

Figure 1 illustrates a HA cluster with two nodes. The HA cluster is managing the availability of a queue manager which has been defined in a resource group. This is an active/passive or cold standby configuration, because only one node, node A, is currently running a queue manager. The queue manager was created with its data and log files on a shared disk. The queue manager has a service IP address which is also managed by the HA cluster. The queue manager depends on the shared disk and its service IP address. When the HA cluster fails the queue manager over from node A to node B, it first moves the queue manager's dependent resources onto node B and then starts the queue manager.

If the HA cluster contains more than one queue manager, your HA cluster configuration might result in two or more queue managers running on the same node after a failover. Each queue manager in the HA cluster must be assigned its own port number, which it uses on whichever cluster node it happens to be active at any particular time.

Generally, the HA cluster runs as the root user. IBM MQ runs as the mqm user. Administration of IBM MQ is granted to members of the mqm group. Ensure that the mqm user and group both exist on all HA cluster nodes. The user ID and group ID must be consistent across the cluster. Administration of IBM MQ by the root user is not allowed; scripts that start, stop, or monitor scripts must switch to the mqm user.

Note: IBM MQ must be installed correctly on all nodes; you cannot share the product executable files.

Configuring the shared disks

An IBM MQ queue manager in an HA cluster requires data files and log files to be in common named remote file systems on a shared disk.

To configure the shared disks, complete the following steps:

1. Decide the names of the mount points for the queue manager's file systems. For example, `/MQHA/qmgrname/data` for the queue manager's data files and `/MQHA/qmgrname/log` for its log files.

2. Create a volume group (or disk group) to contain the queue manager's data and log files. This volume group is managed by the high availability (HA) cluster in the same resource group as the queue manager.
3. Create the file systems for the queue manager's data and log files in the volume group.
4. For each node in turn, create the mount points for the file systems and make sure that the file systems can be mounted. The mqm user must own the mount points.

Figure 1 shows a possible layout for a queue manager in an HA cluster. The queue manager's data and log directories are both on the shared disk which is mounted at /MQHA/QM1. This disk is switched between the nodes of the HA cluster when failover occurs so that the data is available wherever the queue manager is restarted. The `mqs.ini` file has a stanza for the QM1 queue manager. The Log stanza in the `qm.ini` file has a value for LogPath.

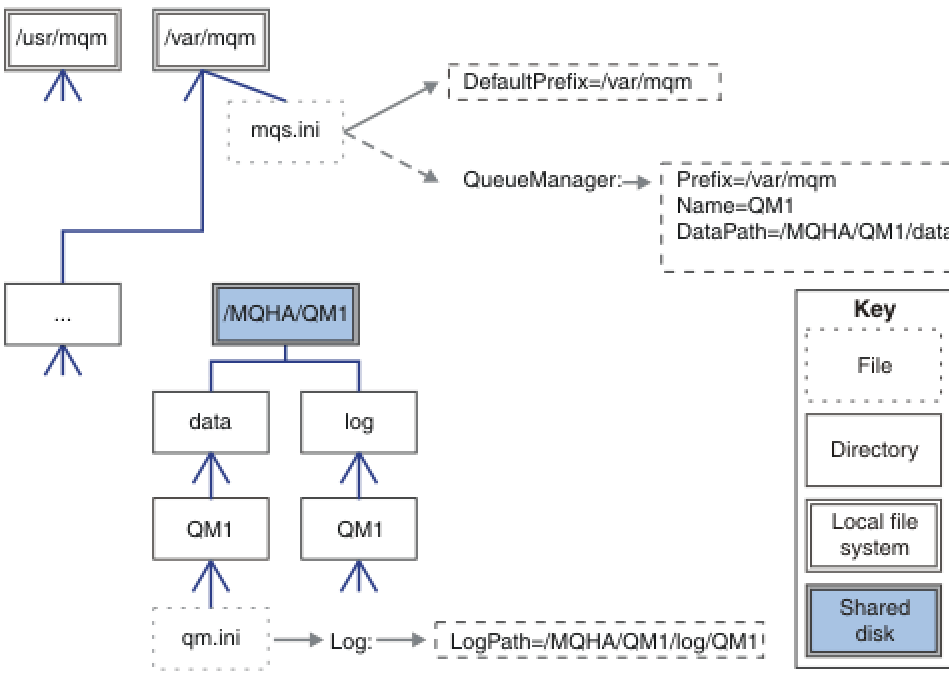

Figure 72. Shared named `data` and `log` directories

Creating a queue manager for use in a high availability (HA) cluster

The first step towards using a queue manager in a high availability cluster is to create the queue manager on one of the nodes.

To create a queue manager for use in an HA cluster, select one of the nodes in the cluster on which to create the queue manager. On this node complete the following steps:

1. Mount the queue manager's file systems on the node.
2. Create the queue manager by using the **crtmqm** command. For example:

   ```
   crtmqm -md /MQHA/qmgrname/data -ld /MQHA/qmgrname/log qmgrname
   ```
3. Start the queue manager manually by using the **strmqm** command.
4. Complete any initial configuration of the queue manager, such as creating queues and channels, and setting the queue manager to start a listener automatically when the queue manager starts.
5. Stop the queue manager by using the **endmqm** command.
6. Use the **dspmqinf** command to display the **addmqinf** command that you can use in a later task, which is documented in "Adding queue manager configuration information to other nodes in a high availability (HA) cluster" on page 382:

   ```
   dspmqinf -o command qmgrname
   ```

   where `qmgrname` is the name of the queue manager.

7. The **addmqinf** command that is displayed will be similar to the following example:

   ```
   addmqinf -sQueueManager -vName=qmgrname -vDirectory=qmgrname \
   -vPrefix=/var/mqm -vDataPath=/MQHA/qmgrname/data/qmgrname
   ```

   Make a careful note of the displayed command.
8. Unmount the queue manager's file systems.

You are now ready to complete the steps described in “Adding queue manager configuration information to other nodes in a high availability (HA) cluster” on page 382.

Adding queue manager configuration information to other nodes in a high availability (HA) cluster

You must add the queue manager configuration to the other nodes in the HA cluster.

Before you complete this task, you must have completed the steps in “Creating a queue manager for use in a high availability (HA) cluster” on page 381.

To add the configuration information for the queue manager to each of other nodes in the HA cluster, complete the following steps on each of the other nodes:

1. Mount the queue manager file systems.
2. Add the queue manager configuration information to the node, either by editing `/var/mqm/mqs.ini` directly, or by issuing the **addmqinf** command that was displayed by the **dspmqinf** command in steps 6 and 7 in “Creating a queue manager for use in a high availability (HA) cluster” on page 381.
3. Start and stop the queue manager to verify the configuration.

   The commands used to start and stop the queue manager must be issued from the same IBM MQ installation as the **addmqinf** command. To start and stop the queue manager from a different installation, you must first set the installation associated with the queue manager using the **setmqm** command. For more information, see setmqm.
4. Unmount the queue manager file systems.

Starting a queue manager under control of a high availability (HA) cluster

The queue manager is represented in the HA cluster as a resource. The HA cluster must be able to start and stop the queue manager. In most cases you can use a shell script to start the queue manager. You must make these scripts available at the same location on all nodes in the cluster, either using a network filesystem or by copying them to each of the local disks.

Note: Before you restart a failed queue manager, you must disconnect your applications from that instance of the queue manager. If you do not, the queue manager might not restart correctly.

Examples of suitable shell scripts are given here. You can tailor these to your needs and use them to start the queue manager under the control of your HA cluster.

The following shell script is an example of how to switch from the HA cluster user to the mqm user so that the queue manager can be successfully started:

```
#!/bin/ksh

# A simple wrapper script to switch to the mqm user.

su mqm -c name_of_your_script $*
```

The following shell script is an example of how to start a queue manager without making any assumptions about the current state of the queue manager. Note that it uses an extremely abrupt method of ending any processes that belong to the queue manager:

```
#!/bin/ksh
#
# This script robustly starts the queue manager.
```

```
#
# The script must be run by the mqm user.

# The only argument is the queue manager name. Save it as QM variable
QM=$1

if [ -z "$QM" ]
then
  echo "ERROR! No queue manager name supplied"
  exit 1
fi

# End any queue manager processes which might be running.

srchstr="( |-m)$QM *.*$"
for process in amqzmuc0 amqzxma0 amqfcxba amqfqpub amqpcsea amqzlaa0 \
               amqzlsa0 runmqchi runmqlsr amqcrsta amqrrmfa amqrmppa \
               amqzfuma amqzmuf0 amqzmur0 amqzmgr0
 do
  ps -ef | tr "\t" " " | grep $process | grep -v grep | \
    egrep "$srchstr" | awk '{print $2}'| \
       xargs kill -9 > /dev/null 2>&1
done

# It is now safe to start the queue manager.
# The strmqm command does not use the -x flag.
strmqm ${QM}
```

You can modify the script to start other related programs.

Stopping a queue manager under the control of a high availability (HA) cluster

In most cases, you can use a shell script to stop a queue manager. Examples of suitable shell scripts are given here. You can tailor these to your needs and use them to stop the queue manager under control of your HA cluster.

The following script is an example of how to immediately stop without making assumptions about the current state of the queue manager. The script must be run by the mqm user. It might therefore be necessary to wrap this script in a shell script to switch the user from the HA cluster user to mqm. (An example shell script is provided in "Starting a queue manager under control of a high availability (HA) cluster" on page 382.)

```
#!/bin/ksh
#
# The script ends the QM by using two phases, initially trying an immediate
# end with a time-out and escalating to a forced stop of remaining
# processes.
#
# The script must be run by the mqm user.
#
# There are two arguments: the queue manager name and a timeout value.
QM=$1
TIMEOUT=$2

if [ -z "$QM" ]
then
  echo "ERROR! No queue manager name supplied"
  exit 1
fi

if [ -z "$TIMEOUT" ]
then
  echo "ERROR! No timeout specified"
  exit 1
fi

for severity in immediate brutal
do
  # End the queue manager in the background to avoid
  # it blocking indefinitely. Run the TIMEOUT timer
  # at the same time to interrupt the attempt, and try a
  # more forceful version. If the brutal version fails,
  # nothing more can be done here.

  echo "Attempting ${severity} end of queue manager '${QM}'"
  case $severity in
```

```
  immediate)
    # Minimum severity of endmqm is immediate which severs connections.
    # HA cluster should not be delayed by clients
    endmqm -i ${QM} &
    ;;

  brutal)
    # This is a forced means of stopping queue manager processes.

    srchstr="( |-m)$QM *.*$"
    for process in amqzmuc0 amqzxma0 amqfcxba amqfqpub amqpcsea amqzlaa0 \
               amqzlsa0 runmqchi runmqlsr amqcrsta amqrrmfa amqrmppa \
               amqzfuma amqzmuf0 amqzmur0 amqzmgr0
    do
      ps -ef | tr "\t" " " | grep $process | grep -v grep | \
         egrep "$srchstr" | awk '{print $2}'| \
            xargs kill -9 > /dev/null 2>&1
    done

  esac

  TIMED_OUT=yes
  SECONDS=0
  while (( $SECONDS < ${TIMEOUT} ))
  do
   TIMED_OUT=yes
   i=0
   while [ $i -lt 5 ]
   do
     # Check for execution controller termination
     srchstr="( |-m)$QM *.*$"
     cnt=`ps -ef | tr "\t" " " | grep amqzxma0 | grep -v grep | \
       egrep "$srchstr" | awk '{print $2}' | wc -l `
     i=`expr $i + 1`
     sleep 1
     if [ $cnt -eq 0 ]
     then
       TIMED_OUT=no
       break
     fi
   done

   if [ ${TIMED_OUT} = "no" ]
   then
     break
   fi

   echo "Waiting for ${severity} end of queue manager '${QM}'"
   sleep 1
  done # timeout loop

  if [ ${TIMED_OUT} = "yes" ]
  then
    continue        # to next level of urgency
  else
    break           # queue manager is ended, job is done
  fi

done # next phase
```

Note: Depending on what processes are running for a specific queue manager, the list of queue manager processes included in this script might either not be a complete list or might include more processes than the processes that are running for that queue manager:

```
for process in amqzmuc0 amqzxma0 amqfcxba amqfqpub amqpcsea amqzlaa0 \
               amqzlsa0 runmqchi runmqlsr amqcrsta amqrrmfa amqrmppa \
               amqzfuma amqzmuf0 amqzmur0 amqzmgr0
```

A process can be included in or excluded from this list based on what feature is configured and what processes are running for a specific queue manager. For a complete list of processes and information about stopping the processes in a specific order, see Stopping a queue manager manually on UNIX and Linux.

Monitoring a queue manager

It is usual to provide a way for the high availability (HA) cluster to monitor the state of the queue manager periodically. In most cases, you can use a shell script for this. Examples of suitable shell scripts are given here. You can tailor these scripts to your needs and use them to make additional monitoring checks specific to your environment.

From IBM WebSphere MQ 7.1, it is possible to have multiple installations of IBM MQ coexisting on a system. For more information about multiple installations, see Multiple installations. If you intend to use the monitoring script across multiple installations, including installations at Version 7.1, or higher, you might need to perform some additional steps. If you have a primary installation, or you are using the script with versions earlier than Version 7.1, you do not need to specify the *MQ_INSTALLATION_PATH* to use the script. Otherwise, the following steps ensure that the *MQ_INSTALLATION_PATH* is identified correctly:

1. Use the **crtmqenv** command from a Version 7.1 installation to identify the correct *MQ_INSTALLATION_PATH* for a queue manager:

   ```
   crtmqenv -m qmname
   ```

   This command returns the correct *MQ_INSTALLATION_PATH* value for the queue manager specified by *qmname*.
2. Run the monitoring script with the appropriate *qmname* and *MQ_INSTALLATION_PATH* parameters.

Note: PowerHA for AIX does not provide a way of supplying a parameter to the monitoring program for the queue manager. You must create a separate monitoring program for each queue manager, that encapsulates the queue manager name. Here is an example of a script used on AIX to encapsulate the queue manager name:

```
#!/bin/ksh
su mqm -c name_of_monitoring_script qmname  MQ_INSTALLATION_PATH
```

where *MQ_INSTALLATION_PATH* is an optional parameter that specifies the path to the installation of IBM MQ that the queue manager *qmname* is associated with.

The following script is not robust to the possibility that **runmqsc** hangs. Typically, HA clusters treat a hanging monitoring script as a failure and are themselves robust to this possibility.

The script does, however, tolerate the queue manager being in the starting state. This is because it is common for the HA cluster to start monitoring the queue manager as soon as it has started it. Some HA clusters distinguish between a starting phase and a running phase for resources, but it is necessary to configure the duration of the starting phase. Because the time taken to start a queue manager depends on the amount of work that it has to do, it is hard to choose a maximum time that starting a queue manager takes. If you choose a value that is too low, the HA cluster incorrectly assumes that the queue manager failed when it has not completed starting. This could result in an endless sequence of failovers.

This script must be run by the mqm user; it might therefore be necessary to wrap this script in a shell script to switch the user from the HA cluster user to mqm (an example shell script is provided in “Starting a queue manager under control of a high availability (HA) cluster” on page 382):

```
#!/bin/ksh
#
# This script tests the operation of the queue manager.
#
# An exit code is generated by the runmqsc command:
# 0  => Either the queue manager is starting or the queue manager is running and responds.
#       Either is OK.
# >0 => The queue manager is not responding and not starting.
#
# This script must be run by the mqm user.
QM=$1
MQ_INSTALLATION_PATH=$2

if [ -z "$QM" ]
then
  echo "ERROR! No queue manager name supplied"
  exit 1
fi
```

```
if [ -z "$MQ_INSTALLATION_PATH" ]
then
  # No path specified, assume system primary install or MQ level < 7.1.0.0
  echo "INFO: Using shell default value for MQ_INSTALLATION_PATH"
else
  echo "INFO: Prefixing shell PATH variable with $MQ_INSTALLATION_PATH/bin"
  PATH=$MQ_INSTALLATION_PATH/bin:$PATH
fi

# Test the operation of the queue manager. Result is 0 on success, non-zero on error.
echo "ping qmgr" | runmqsc ${QM} > /dev/null 2>&1
pingresult=$?

if [ $pingresult -eq 0 ]
then # ping succeeded

  echo "Queue manager '${QM}' is responsive"
  result=0

else # ping failed

  # Don't condemn the queue manager immediately, it might be starting.
  srchstr="( |-m)$QM *.*$"
  cnt=`ps -ef | tr "\t" " " | grep strmqm | grep "$srchstr" | grep -v grep \
                | awk '{print $2}' | wc -l`
  if [ $cnt -gt 0 ]
  then
    # It appears that the queue manager is still starting up, tolerate
    echo "Queue manager '${QM}' is starting"
    result=0
  else
    # There is no sign of the queue manager starting
    echo "Queue manager '${QM}' is not responsive"
    result=$pingresult
  fi

fi

exit $result
```

Putting the queue manager under control of the high availability (HA) cluster

You must configure the queue manager, under control of the HA cluster, with the queue manager's IP address and shared disks.

To define a resource group to contain the queue manager and all of its associated resources, complete the following steps:

1. Create the resource group containing the queue manager, the queue manager's volume or disk group, and the queue manager's IP address. The IP address is a virtual IP address, not the IP address of the computer.
2. Verify that the HA cluster correctly switches the resources between the cluster nodes and is ready to control the queue manager.

Deleting a queue manager from a high availability (HA) cluster node

You might want to remove a queue manager from a node that is no longer required to run the queue manager.

To remove the queue manager from a node in an HA cluster, complete the following steps:

1. Remove the node from the HA cluster so that the HA cluster will no longer attempt to activate the queue manager on this node.
2. Use the following **rmvmqinf** command to remove the queue manager's configuration information:

   `rmvmqinf` *qmgrname*

To completely delete the queue manager, use the **dltmqm** command. However, be aware that this completely deletes the queue manager's data and log files. When you have deleted the queue manager, you can use the **rmvmqinf** command to remove remaining configuration information from the other nodes.

Supporting the Microsoft Cluster Service (MSCS)

Introducing and setting up MSCS to support failover of virtual servers.

This information applies to IBM MQ for Windows only.

The Microsoft Cluster Service (MSCS) enables you to connect servers into a *cluster*, giving higher availability of data and applications, and making it easier to manage the system. MSCS can automatically detect and recover from server or application failures.

MSCS supports *failover* of *virtual servers*, which correspond to applications, Web sites, print queues, or file shares (including, for example, their disk spindles, files, and IP addresses).

Failover is the process by which MSCS detects a failure in an application on one computer in the cluster, and shuts down the disrupted application in an orderly manner, transfers its state data to the other computer, and reinitiates the application there.

This section introduces MSCS clusters and describes setting up MSCS support in the following sections:

- “Introducing MSCS clusters” on page 387
- “Setting up IBM MQ for MSCS clustering” on page 388

Then tells you how to configure IBM MQ for MSCS clustering, in the following sections:

- “Creating a queue manager for use with MSCS” on page 390
- “Moving a queue manager to MSCS storage” on page 391
- “Putting a queue manager under MSCS control” on page 392
- “Removing a queue manager from MSCS control” on page 398

And then gives some useful hints on using MSCS with IBM MQ, and details the IBM MQ MSCS support utility programs, in the following sections:

- “Hints and tips on using MSCS” on page 399
- “IBM MQ MSCS support utility programs” on page 401

Introducing MSCS clusters

MSCS clusters are groups of two or more computers, connected together and configured in such a way that, if one fails, MSCS performs a *failover*, transferring the state data of applications from the failing computer to another computer in the cluster and re-initiating their operation there.

“Using IBM MQ with high availability configurations” on page 377 contains a comparison between MSCS clusters, multi-instance queue managers, and IBM MQ clusters.

In this section and its subordinate topics, the term *cluster*, when used by itself, **always** means an MSCS cluster. This is distinct from an IBM MQ cluster described elsewhere in this guide.

A two-machine cluster comprises two computers (for example, A and B) that are jointly connected to a network for client access using a *virtual IP address*. They might also be connected to each other by one or more private networks. A and B share at least one disk for the server applications on each to use. There is also another shared disk, which must be a redundant array of independent disks (*RAID*) Level 1, for the exclusive use of MSCS; this is known as the *quorum* disk. MSCS monitors both computers to check that the hardware and software are running correctly.

In a simple setup such as this, both computers have all the applications installed on them, but only computer A runs with live applications; computer B is just running and waiting. If computer A encounters any one of a range of problems, MSCS shuts down the disrupted application in an orderly manner, transfers its state data to the other computer, and re-initiates the application there. This is known as a *failover*. Applications can be made *cluster-aware* so that they interact fully with MSCS and failover gracefully.

A typical setup for a two-computer cluster is as shown in Figure 73 on page 388.

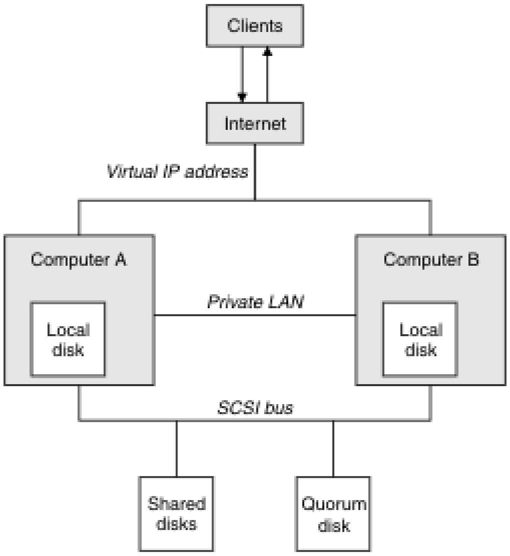

Figure 73. Two-computer MSCS cluster

Each computer can access the shared disk, but only one at a time, under the control of MSCS. In the event of a failover, MSCS switches the access to the other computer. The shared disk itself is usually a RAID, but need not be.

Each computer is connected to the external network for client access, and each has an IP address. However an external client, communicating with this cluster, is aware of only one *virtual IP address*, and MSCS routes the IP traffic within the cluster appropriately.

MSCS also performs its own communications between the two computers, either over one or more private connections or over the public network, for example to monitor their states using the heartbeat, and to synchronize their databases.

Setting up IBM MQ for MSCS clustering

You configure IBM MQ for clustering by making the queue manager the unit of failover to MSCS. You define a queue manager as a resource to MSCS, which can then monitor it, and transfer it to another computer in the cluster if there is a problem.

To set your system up for this, you start by installing IBM MQ on each computer in the cluster.

As the queue manager is associated with the IBM MQ installation name, the IBM MQ installation name on all the computers in the cluster should be the same. See Installing and uninstalling.

The queue managers themselves need to exist only on the computer on which you create them. In the event of a failover, the MSCS initiates the queue managers on the other computer. The queue managers, however, must have their log and data files on a cluster shared disk, and not on a local drive. If you have a queue manager already installed on a local drive, you can migrate it using a tool provided with IBM MQ; see “Moving a queue manager to MSCS storage” on page 391. If you want to create new queue managers for use with MSCS, see “Creating a queue manager for use with MSCS” on page 390.

After installation and migration, use the MSCS Cluster Administrator to make MSCS aware of your queue managers; see “Putting a queue manager under MSCS control” on page 392.

If you decide to remove a queue manager from MSCS control, use the procedure described in “Removing a queue manager from MSCS control” on page 398.

Setup symmetry

When an application switches from one node to the other it must behave in the same way, regardless of node. The best way of ensuring this is to make the environments identical.

If you can, set up a cluster with identical hardware, operating system software, product software, and configuration on each computer. In particular, ensure that all the required software installed on the two computers is identical in terms of version, maintenance level, SupportPacs, paths and exits, and that there is a common namespace (security environment) as described in "MSCS security" on page 389.

MSCS security
For successful MSCS security, follow these guidelines.

The guidelines are as follows:

- Make sure you that you have identical software installations on each computer in the cluster.
- Create a common namespace (security environment) across the cluster.
- Make the nodes of the MSCS cluster members of a domain, within which the user account that is the *cluster owner* is a domain account.
- Make the other user accounts on the cluster also domain accounts, so that they are available on both nodes. This is automatically the case if you already have a domain, and the accounts relevant to IBM MQ are domain accounts. If you do not currently have a domain, consider setting up a *mini-domain* to cater for the cluster nodes and relevant accounts. Your aim is to make your cluster of two computers look like a single computing resource.

  Remember that an account that is local to one computer does not exist on the other one. Even if you create an account with the same name on the other computer, its security identifier (SID) is different, so, when your application is moved to the other node, the permissions do not exist on that node.

During a failover or move, IBM MQ MSCS support ensures that all files that contain queue manager objects have equivalent permissions on the destination node. Explicitly, the code checks that the Administrators and mqm groups, and the SYSTEM account, have full control, and that if `Everyone` had read access on the old node, that permission is added on the destination node.

You can use a domain account to run your IBM MQ Service. Make sure that it exists in the local mqm group on each computer in the cluster.

Using multiple queue managers with MSCS
If you are running more than one queue manager on a computer, you can choose one of these setups.

The setups are as follows:

- All the queue managers in a single group. In this configuration, if a problem occurs with any queue manager, all the queue managers in the group failover to the other computer as a group.
- A single queue manager in each group. In this configuration, if a problem occurs with the queue manager, it alone fails over to the other computer without affecting the other queue managers.
- A mixture of the first two setups.

Cluster modes
There are two modes in which you might run a cluster system with IBM MQ: Active/Passive or Active/Active.

Note: If you are using MSCS together with the Microsoft Transaction Server (COM+), you cannot use Active/Active mode.

Active/Passive mode

In Active/Passive mode, computer A has the running application on it, and computer B is backup, only being used when MSCS detects a problem.

You can use this mode with only one shared disk, but, if any application causes a failover, **all** the applications must be transferred as a group (because only one computer can access the shared disk at a time).

You can configure MSCS with A as the *preferred* computer. Then, when computer A has been repaired or replaced and is working properly again, MSCS detects this and automatically switches the application back to computer A.

If you run more than one queue manager, consider having a separate shared disk for each. Then put each queue manager in a separate group in MSCS. In this way, any queue manager can failover to the other computer without affecting the other queue managers.

Active/Active mode

In Active/Active mode, computers A and B both have running applications, and the groups on each computer are set to use the other computer as backup. If a failure is detected on computer A, MSCS transfers the state data to computer B, and reinitiates the application there. computer B then runs its own application and A's.

For this setup you need at least two shared disks. You can configure MSCS with A as the preferred computer for A's applications, and B as the preferred computer for B's applications. After failover and repair, each application automatically ends up back on its own computer.

For IBM MQ this means that you could, for example, run two queue managers, one on each of A and B, with each exploiting the full power of its own computer. After a failure on computer A, both queue managers will run on computer B. This will mean sharing the power of the one computer, with a reduced ability to process large quantities of data at speed. However, your critical applications will still be available while you find and repair the fault on A.

Creating a queue manager for use with MSCS

This procedure ensures that a new queue manager is created in such a way that it is suitable for preparing and placing under MSCS control.

You start by creating the queue manager with all its resources on a local drive, and then migrate the log files and data files to a shared disk. (You can reverse this operation.) Do **not** attempt to create a queue manager with its resources on a shared drive.

You can create a queue manager for use with MSCS in two ways, either from a command prompt, or in the IBM MQ Explorer. The advantage of using a command prompt is that the queue manager is created *stopped* and set to *manual startup*, which is ready for MSCS. (The IBM MQ Explorer automatically starts a new queue manager and sets it to automatic startup after creation. You have to change this.)

Creating a queue manager from a command prompt

Follow these steps to create a queue manager from a command prompt, for use with MSCS:

1. Ensure that you have the environment variable `MQSPREFIX` set to refer to a local drive, for example `C:\WebSphere MQ`. If you change this, reboot the machine so that the System account picks up the change. If you do not set the variable, the queue manager is created in the IBM MQ default directory for queue managers.
2. Create the queue manager using the **crtmqm** command. For example, to create a queue manager called `mscs_test` in the default directory, use:

   ```
   crtmqm mscs_test
   ```

3. Proceed to "Moving a queue manager to MSCS storage" on page 391.

Creating a queue manager using the IBM MQ Explorer

Follow these steps to create a queue manager using the IBM MQ Explorer, for use with MSCS:

1. Start the IBM MQ Explorer from the Start menu.
2. In the Navigator View, expand the tree nodes to find the `Queue Managers` tree node.
3. Right-click the `Queue Managers` tree node, and select `New` -> `Queue Manager`. The Create Queue Manager panel is displayed.
4. Complete the dialog (Step 1), then click `Next>`.

5. Complete the dialog (Step 2), then click Next> .
6. Complete the dialog (Step 3), ensuring that Start Queue Manager and Create Server Connection Channel are not selected, then click Next> .
7. Complete the dialog (Step 4), then click Finish .
8. Proceed to “Moving a queue manager to MSCS storage” on page 391.

Moving a queue manager to MSCS storage

This procedure configures an existing queue manager to make it suitable for putting under MSCS control.

To achieve this, you move the log files and data files to shared disks to make them available to the other computer in the event of a failure. For example, the existing queue manager might have paths such as C:\WebSphere MQ\log\<QMname> and C:\WebSphere MQ\qmgrs\<QMname>. Do ***not*** try to move the files by hand; use the utility program supplied as part of IBM MQ MSCS Support as described in this topic.

If the queue manager being moved uses SSL connections and the SSL key repository is in the queue manager data directory on the local machine, then the key repository will be moved with the rest of the queue manager to the shared disk. By default, the queue manager attribute that specifies the SSL key repository location, SSLKEYR, is set to *MQ_INSTALLATION_PATH*\qmgrs\QMGRNAME\ssl\key, which is under the queue manager data directory. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed. The hamvmqm command does not modify this queue manager attribute. In this situation you must modify the queue manager attribute, SSLKEYR, using the IBM MQ Explorer or the MQSC command ALTER QMGR, to point to the new SSL key repository file.

The procedure is as follows:

1. Shut down the queue manager, and check that there are no errors.
2. If the queue manager's log files or queue files are already stored on a shared disk, skip the rest of this procedure and proceed directly to “Putting a queue manager under MSCS control” on page 392.
3. Make a full media backup of the queue files and log files and store the backup in a safe place (see “Queue manager log files” on page 400 for why this is important).
4. If you already have a suitable shared disk resource proceed to step 6. Otherwise, using the MSCS Cluster Administrator to create a resource of type *shared disk* with sufficient capacity to store the queue manager log files and data (queue) files.
5. Test the shared disk by using the MSCS Cluster Administrator to move it from one cluster node to the other and back again.
6. Make sure that the shared disk is online on the cluster node where the queue manager log and data files are stored locally.
7. Run the utility program to move the queue manager as follows:

   ```
   hamvmqm /m qmname /dd " e: \
   IBM MQ " /ld " e: \
   IBM MQ \log"
   ```

   substituting your queue manager name for *qmname*, your shared disk drive letter for *e*, and your chosen directory for *IBM MQ*. The directories are created if they do not already exist.
8. Test the queue manager to ensure that it works, using the IBM MQ Explorer. For example:
   a. Right-click the queue manager tree node, then select Start . The queue manager starts.
   b. Right-click the Queues tree node, then select New -> Local Queue... , and give the queue a name.
   c. Click Finish .
   d. Right-click the queue, then select Put Test Message... . The Put Test Message panel is displayed.
   e. Type some message text, then click Put Test Message , and close the panel.

f. Right-click the queue, then select `Browse Messages...`. The Message Browser panel is displayed.
g. Ensure your message is on the queue, then click `Close`. The Message Browser panel closes.
h. Right-click the queue, then select `Clear Messages...`. The messages on the queue are cleared.
i. Right-click the queue, then select `Delete...`. A confirmation panel is displayed, click `OK`. The queue is deleted.
j. Right-click the queue manager tree node, then select `Stop...`. The End Queue Manager panel is displayed.
k. Click `OK`. The queue manager stops.

9. As IBM MQ Administrator ensure that the startup attribute of the queue manager is set to manual. In the IBM MQ Explorer, set the Startup field to `manual` in the queue manager properties panel.
10. Proceed to "Putting a queue manager under MSCS control" on page 392.

Putting a queue manager under MSCS control

The tasks involved in placing a queue manager under MSCS control, including prerequisite tasks.

Before you put a queue manager under MSCS control

Before you put a queue manager under MSCS control, perform the following tasks:

1. Ensure that IBM MQ and its MSCS Support are installed on both machines in the cluster and that the software on each computer is identical, as described in "Setting up IBM MQ for MSCS clustering" on page 388.
2. Use the **haregtyp** utility program to register IBM MQ as an MSCS resource type on all the cluster nodes. See "IBM MQ MSCS support utility programs" on page 401 for additional information.
3. If you have not yet created the queue manager, see "Creating a queue manager for use with MSCS" on page 390.
4. If you have created the queue manager, or it already exists, ensure that you have carried out the procedure in "Moving a queue manager to MSCS storage" on page 391.
5. Stop the queue manager, if it is running, using either a command prompt or the IBM MQ Explorer.
6. Test MSCS operation of the shared drives before going on to either of the following Windows procedures in this topic.

Windows Server 2012

To place a queue manager under MSCS control on Windows Server 2012, use the following procedure:

1. Log in to the cluster node computer hosting the queue manager, or log in to a remote workstation as a user with cluster administration permissions, and connect to the cluster node hosting the queue manager.
2. Start the Failover Cluster Management tool.
3. Right-click **Failover Cluster Management > Connect Cluster ...** to open a connection to the cluster.
4. In contrast to the group scheme used in the MSCS Cluster Administrator on previous versions of Windows, the Failover Cluster Management tool uses the concept of services and applications. A configured service or application contains all the resources necessary for one application to be clustered. You can configure a queue manager under MSCS as follows:

    a. Right-click on the cluster and select **Configure Role** to start the configuration wizard.
    b. Select **Other Server** on the "Select Service or Application" panel.
    c. Select an appropriate IP address as a client access point.

       This address should be an unused IP address to be used by clients and other queue managers to connect to the *virtual* queue manager. This IP address is not the normal (static) address of either node; it is an additional address that *floats* between them. Although MSCS handles the routing of this address, it does **not** verify that the address can be reached.

d. Assign a storage device for exclusive use by the queue manager. This device needs to be created as a resource instance before it can be assigned.

   You can use one drive to store both the logs and queue files, or you can split them up across drives. In either case, if each queue manager has its own shared disk, ensure that all drives used by this queue manager are exclusive to this queue manager, that is, that nothing else relies on the drives. Also ensure that you create a resource instance for every drive that the queue manager uses.

   The resource type for a drive depends on the SCSI support you are using; refer to your SCSI adapter instructions. There might already be groups and resources for each of the shared drives. If so, you do not need to create the resource instance for each drive. Move it from its current group to the one created for the queue manager.

   For each drive resource, set possible owners to both nodes. Set dependent resources to none.

e. Select the **MQSeries MSCS** resource on the "Select Resource Type" panel.

f. Complete the remaining steps in the wizard.

5. Before bringing the resource online, the MQSeries® MSCS resource needs additional configuration:

   a. Select the newly defined service which contains a resource called 'New MQSeries MSCS'.

   b. Right-click **Properties** on the MQ resource.

   c. Configure the resource:

      - `Name` ; choose a name that makes it easy to identify which queue manager it is for.
      - `Run in a separate Resource Monitor` ; for better isolation
      - `Possible owners` ; set both nodes
      - `Dependencies` ; add the drive and IP address for this queue manager.

        Warning: Failure to add these dependencies means that IBM MQ attempts to write the queue manager status to the wrong cluster disk during failovers. Because many processes might be attempting to write to this disk simultaneously, some IBM MQ processes could be blocked from running.

      - `Parameters` ; as follows:

        – `QueueManagerName` (required); the name of the queue manager that this resource is to control. This queue manager must exist on the local computer.

        – `PostOnlineCommand` (optional); you can specify a program to run whenever the queue manager resource changes its state from offline to online. For more details see "PostOnlineCommand and PreOfflineCommand" on page 401.

        – `PreOfflineCommand` (optional); you can specify a program to run whenever the queue manager resource changes its state from online to offline. For more details see "PostOnlineCommand and PreOfflineCommand" on page 401.

        Note: The *looksAlive* poll interval is set to default value of 5000 ms. The *isAlive* poll interval is set to default value of 60000 ms. These defaults can only be modified after the resource definition has been completed. For further details see "Summary of looksAlive and isAlive polling" on page 397.

   d. Optionally, set a preferred node (but note the comments in "Using preferred nodes" on page 401)

   e. The *Failover Policy* is set by default to sensible values, but you can tune the thresholds and periods that control *Resource Failover* and *Group Failover* to match the loads placed on the queue manager.

6. Test the queue manager by bringing it online in the MSCS Cluster Administrator and subjecting it to a test workload. If you are experimenting with a test queue manager, use the IBM MQ Explorer. For example:

   a. Right-click the `Queues` tree node, then select `New -> Local Queue...` , and give the queue a name.

b. Click `Finish` . The queue is created, and displayed in the content view.
c. Right-click the queue, then select `Put Test Message...` . The Put Test Message panel is displayed.
d. Type some message text, then click `Put Test Message` , and close the panel.
e. Right-click the queue, then select `Browse Messages...` . The Message Browser panel is displayed.
f. Ensure that your message is on the queue, then click `Close` . The Message Browser panel closes.
g. Right-click the queue, then select `Clear Messages...` . The messages on the queue are cleared.
h. Right-click the queue, then select `Delete...` . A confirmation panel is displayed, click `OK` . The queue is deleted.

7. Test that the queue manager can be taken offline and back online using the MSCS Cluster Administrator.
8. Simulate a failover.

   In the MSCS Cluster Administrator, right-click the group containing the queue manager and select `Move Group`. This can take some minutes to do. (If at other times you want to move a queue manager to another node quickly, follow the procedure in "Moving a queue manager to MSCS storage" on page 391.) You can also right-click and select `Initiate Failure` ; the action (local restart or failover) depends on the current state and the configuration settings.

Windows Server 2008

To place a queue manager under MSCS control on Windows Server 2008, use the following procedure:

1. Log in to the cluster node computer hosting the queue manager, or log in to a remote workstation as a user with cluster administration permissions, and connect to the cluster node hosting the queue manager.
2. Start the Failover Cluster Management tool.
3. Right-click **Failover Cluster Management > Manage a Cluster ...** to open a connection to the cluster.
4. In contrast to the group scheme used in the MSCS Cluster Administrator on previous versions of Windows, the Failover Cluster Management tool uses the concept of services and applications. A configured service or application contains all the resources necessary for one application to be clustered. You can configure a queue manager under MSCS as follows:

   a. Right-click **Services and Applications > Configure a Service or Application ...** to start the configuration wizard.
   b. Select **Other Server** on the **Select Service or Application** panel.
   c. Select an appropriate IP address as a client access point.

      This address should be an unused IP address to be used by clients and other queue managers to connect to the *virtual* queue manager. This IP address is not the normal (static) address of either node; it is an additional address that *floats* between them. Although MSCS handles the routing of this address, it does **not** verify that the address can be reached.
   d. Assign a storage device for exclusive use by the queue manager. This device needs to be created as a resource instance before it can be assigned.

      You can use one drive to store both the logs and queue files, or you can split them up across drives. In either case, if each queue manager has its own shared disk, ensure that all drives used by this queue manager are exclusive to this queue manager, that is, that nothing else relies on the drives. Also ensure that you create a resource instance for every drive that the queue manager uses.

      The resource type for a drive depends on the SCSI support you are using; refer to your SCSI adapter instructions. There might already be groups and resources for each of the shared drives. If so, you do not need to create the resource instance for each drive. Move it from its current group to the one created for the queue manager.

For each drive resource, set possible owners to both nodes. Set dependent resources to none.

e. Select the **MQSeries MSCS** resource on the **Select Resource Type** panel.

f. Complete the remaining steps in the wizard.

5. Before bringing the resource online, the MQSeries MSCS resource needs additional configuration:

   a. Select the newly defined service which contains a resource called 'New MQSeries MSCS'.

   b. Right-click **Properties** on the MQ resource.

   c. Configure the resource:

      - `Name` ; choose a name that makes it easy to identify which queue manager it is for.
      - `Run in a separate Resource Monitor` ; for better isolation
      - `Possible owners` ; set both nodes
      - `Dependencies` ; add the drive and IP address for this queue manager.

        Warning: Failure to add these dependencies means that IBM MQ attempts to write the queue manager status to the wrong cluster disk during failovers. Because many processes might be attempting to write to this disk simultaneously, some IBM MQ processes could be blocked from running.
      - `Parameters` ; as follows:
        - `QueueManagerName` (required); the name of the queue manager that this resource is to control. This queue manager must exist on the local computer.
        - `PostOnlineCommand` (optional); you can specify a program to run whenever the queue manager resource changes its state from offline to online. For more details see "PostOnlineCommand and PreOfflineCommand" on page 401.
        - `PreOfflineCommand` (optional); you can specify a program to run whenever the queue manager resource changes its state from online to offline. For more details see "PostOnlineCommand and PreOfflineCommand" on page 401.

          Note: The *looksAlive* poll interval is set to default value of 5000 ms. The *isAlive* poll interval is set to default value of 60000 ms. These defaults can only be modified after the resource definition has been completed. For further details see "Summary of looksAlive and isAlive polling" on page 397.

   d. Optionally, set a preferred node (but note the comments in "Using preferred nodes" on page 401)

   e. The *Failover Policy* is set by default to sensible values, but you can tune the thresholds and periods that control *Resource Failover* and *Group Failover* to match the loads placed on the queue manager.

6. Test the queue manager by bringing it online in the MSCS Cluster Administrator and subjecting it to a test workload. If you are experimenting with a test queue manager, use the IBM MQ Explorer. For example:

   a. Right-click the `Queues` tree node, then select `New` -> `Local Queue...` , and give the queue a name.

   b. Click `Finish` . The queue is created, and displayed in the content view.

   c. Right-click the queue, then select `Put Test Message...` . The **Put Test Message** panel is displayed.

   d. Type some message text, then click `Put Test Message` , and close the panel.

   e. Right-click the queue, then select `Browse Messages...` . The **Message Browser** panel is displayed.

   f. Ensure that your message is on the queue, then click `Close` . The **Message Browser** panel closes.

   g. Right-click the queue, then select `Clear Messages...` . The messages on the queue are cleared.

   h. Right-click the queue, then select `Delete...` . A confirmation panel is displayed, click `OK` . The queue is deleted.

7. Test that the queue manager can be taken offline and back online using the MSCS Cluster Administrator.
8. Simulate a failover.

   In the MSCS Cluster Administrator, right-click the group containing the queue manager and select `Move Group`. This can take some minutes to do. (If at other times you want to move a queue manager to another node quickly, follow the procedure in "Moving a queue manager to MSCS storage" on page 391.) You can also right-click and select `Initiate Failure` ; the action (local restart or failover) depends on the current state and the configuration settings.

Windows 2003

To place a queue manager under MSCS control on Windows 2003, use the following procedure:

1. Log in to the cluster node computer hosting the queue manager, or log in to a remote workstation as a user with cluster administration permissions, and connect to the cluster node hosting the queue manager.
2. Start the MSCS Cluster Administrator.
3. Open a connection to the cluster.
4. Create an MSCS group to be used to contain the resources for the queue manager. Name the group in such a way that it is obvious which queue manager it relates to. Each group can contain multiple queue managers, as described in "Using multiple queue managers with MSCS" on page 389.

   Use the group for all the remaining steps.
5. Create a resource instance for each of the SCSI logical drives that the queue manager uses.

   You can use one drive to store both the logs and queue files, or you can split them up across drives. In either case, if each queue manager has its own shared disk, ensure that all drives used by this queue manager are exclusive to this queue manager, that is, that nothing else relies on the drives. Also ensure that you create a resource instance for every drive that the queue manager uses.

   The resource type for a drive depends on the SCSI support you are using; refer to your SCSI adapter instructions. There might already be groups and resources for each of the shared drives. If so, you do not need to create the resource instance for each drive. Move it from its current group to the one created for the queue manager.

   For each drive resource, set possible owners to both nodes. Set dependent resources to none.
6. Create a resource instance for the IP address.

   Create an IP address resource (resource type *IP address*). This address should be an unused IP address to be used by clients and other queue managers to connect to the *virtual* queue manager. This IP address is not the normal (static) address of either node; it is an additional address that *floats* between them. Although MSCS handles the routing of this address, it does **not** verify that the address can be reached.
7. Create a resource instance for the queue manager.

   Create a resource of type *IBM MQ MSCS*. The wizard prompts you for various items, including the following:

   - `Name` ; choose a name that makes it easy to identify which queue manager it is for.
   - `Add to group` ; use the group that you created
   - `Run in a separate Resource Monitor` ; for better isolation
   - `Possible owners` ; set both nodes
   - `Dependencies` ; add the drive and IP address for this queue manager.

     Warning: Failure to add these dependencies means that IBM MQ attempts to write the queue manager status to the wrong cluster disk during failovers. Because many processes might be attempting to write to this disk simultaneously, some IBM MQ processes could be blocked from running.

- `Parameters` ; as follows:
  - `QueueManagerName` (required); the name of the queue manager that this resource is to control. This queue manager must exist on the local computer.
  - `PostOnlineCommand` (optional); you can specify a program to run whenever the queue manager resource changes its state from offline to online. For more details see "PostOnlineCommand and PreOfflineCommand" on page 401.
  - `PreOfflineCommand` (optional); you can specify a program to run whenever the queue manager resource changes its state from online to offline. For more details see "PostOnlineCommand and PreOfflineCommand" on page 401.

    Note: The *looksAlive* poll interval is set to default value of 5000 ms. The *isAlive* poll interval is set to default value of 30000 ms. These defaults can only be modified after the resource definition has been completed. For further details see "Summary of looksAlive and isAlive polling" on page 397.

8. Optionally, set a preferred node (but note the comments in "Using preferred nodes" on page 401)
9. The *Failover Policy* (as defined in the properties for the group) is set by default to sensible values, but you can tune the thresholds and periods that control *Resource Failover* and *Group Failover* to match the loads placed on the queue manager.
10. Test the queue manager by bringing it online in the MSCS Cluster Administrator and subjecting it to a test workload. If you are experimenting with a test queue manager, use the IBM MQ Explorer. For example:
    a. Right-click the `Queues` tree node, then select `New` -> `Local Queue...` , and give the queue a name.
    b. Click `Finish` . The queue is created, and displayed in the content view.
    c. Right-click the queue, then select `Put Test Message...` . The **Put Test Message** panel is displayed.
    d. Type some message text, then click `Put Test Message` , and close the panel.
    e. Right-click the queue, then select `Browse Messages...` . The **Message Browser** panel is displayed.
    f. Ensure that your message is on the queue, then click `Close` . The **Message Browser** panel closes.
    g. Right-click the queue, then select `Clear Messages...` . The messages on the queue are cleared.
    h. Right-click the queue, then select `Delete...` . A confirmation panel is displayed, click `OK` . The queue is deleted.
11. Test that the queue manager can be taken offline and back online using the MSCS Cluster Administrator.
12. Simulate a failover.

    In the MSCS Cluster Administrator, right-click the group containing the queue manager and select `Move Group`. This can take some minutes to do. (If at other times you want to move a queue manager to another node quickly, follow the procedure in "Moving a queue manager to MSCS storage" on page 391.) You can also right-click and select `Initiate Failure` ; the action (local restart or failover) depends on the current state and the configuration settings.

Summary of looksAlive and isAlive polling

looksAlive and *isAlive* are intervals at which MSCS calls back into the resource types supplied library code and requests that the resource performs checks to determine the working status of itself. This ultimately determines if MSCS attempts to fail over the resource.

On every occasion that the *looksAlive* interval elapses (default 5000 ms), the queue manager resource is called to perform its own check to determine if its status is satisfactory.

On every occasion that the *isAlive* interval elapses (default 30000 ms), another call is made to the queue manager resource for it to perform another check to determine if the resource is functioning correctly. This enables two levels of resource type checking.

1. A *looksAlive* status check to establish if the resource appears to be functioning.
2. A more significant *isAlive* check that determines if the queue manager resource is active.

If the queue manager resource is determined not to be active, MSCS, based on other advanced MSCS options, triggers a fail over for the resource and associated dependant resources to another node in the cluster. For further information, see MSCS documentation.

Removing a queue manager from MSCS control

You can remove queue managers from MSCS control, and return them to manual administration.

You do not need to remove queue managers from MSCS control for maintenance operations. You can do that by taking a queue manager offline temporarily, using the MSCS Cluster Administrator. Removing a queue manager from MSCS control is a more permanent change; only do it if you decide that you no longer want MSCS to have any further control of the queue manager.

If the queue manager is being removed uses TSL or SSL connections you must modify the queue manager attribute, SSLKEYR, using the IBM MQ Explorer or the MQSC command `ALTER QMGR`, to point to the SSL key repository file on the local directory.

The procedure is:

1. Take the queue manager resource offline using the MSCS Cluster Administrator, as described in "Taking a queue manager offline from MSCS" on page 398
2. Destroy the resource instance. This does not destroy the queue manager.
3. Optionally, migrate the queue manager files back from shared drives to local drives. To do this, see "Returning a queue manager from MSCS storage" on page 398.
4. Test the queue manager.

Taking a queue manager offline from MSCS

To take a queue manager offline from MSCS, perform the following steps:

1. Start the MSCS Cluster Administrator.
2. Open a connection to the cluster.
3. Select `Groups`, or `Role` if you are using Windows 2012, and open the group containing the queue manager to be moved.
4. Select the queue manager resource.
5. Right-click it and select `Offline`.
6. Wait for completion.

Returning a queue manager from MSCS storage

This procedure configures the queue manager to be back on its computer's local drive, that is, it becomes a *normal* IBM MQ queue manager. To achieve this, you move the log files and data files from the shared disks. For example, the existing queue manager might have paths such as `E:\WebSphere MQ\log\<QMname>` and `E:\WebSphere MQ\qmgrs\<QMname>`. Do not try to move the files by hand; use the **hamvmqm** utility program supplied as part of IBM MQ MSCS Support:

1. Make a full media backup of the queue files and log files and store the backup in a safe place (see "Queue manager log files" on page 400 for why this is important).
2. Decide which local drive to use and ensure that it has sufficient capacity to store the queue manager log files and data (queue) files.
3. Make sure that the shared disk on which the files currently reside is online on the cluster node to which to move the queue manager log and data files.

4. Run the utility program to move the queue manager as follows:

```
hamvmqm /m qmname /dd " c:\
IBM MQ " /ld "c:\
IBM MQ \log"
```

substituting your queue manager name for *qmname*, your local disk drive letter for *c*, and your chosen directory for *IBM MQ* (the directories are created if they do not already exist).

5. Test the queue manager to ensure that it works (as described in "Moving a queue manager to MSCS storage" on page 391).

Hints and tips on using MSCS

This section contains some general information to help you use IBM MQ support for MSCS effectively.

This section contains some general information to help you use IBM MQ support for MSCS effectively.

How long does it take to fail a queue manager over from one machine to the other? This depends heavily on the amount of workload on the queue manager and on the mix of traffic, for example, how much of it is persistent, within sync point, and how much committed before the failure. IBM tests have given failover and failback times of about a minute. This was on a very lightly loaded queue manager and actual times will vary considerably depending on load.

Verifying that MSCS is working
Follow these steps to ensure that you have a running MSCS cluster.

The task descriptions starting with "Creating a queue manager for use with MSCS" on page 390 assume that you have a running MSCS cluster within which you can create, migrate, and destroy resources. If you want to make sure that you have such a cluster:

1. Using the MSCS Cluster Administrator, create a group.
2. Within that group, create an instance of a generic application resource, specifying the system clock (path name `C:\winnt\system32\clock.exe` and working directory of `C:\`).
3. Make sure that you can bring the resource online, that you can move the group that contains it to the other node, and that you can take the resource offline.

Manual startup
For a queue manager managed by MSCS, you ***must*** set the startup attribute to manual. This ensures that the IBM MQ MSCS support can restart the MQSeries Service without immediately starting the queue manager.

The IBM MQ MSCS support needs to be able to restart the service so that it can perform monitoring and control, but must itself remain in control of which queue managers are running, and on which machines. See "Moving a queue manager to MSCS storage" on page 391 for more information.

MSCS and queue managers
Considerations concerning queue managers when using MSCS.

Creating a matching queue manager on the other node

For clustering to work with IBM MQ, you need an identical queue manager on node B for each one on node A. However, you do not need to explicitly create the second one. You can create or prepare a queue manager on one node, move it to the other node as described in "Moving a queue manager to MSCS storage" on page 391, and it is fully duplicated on that node.

Default queue managers

Do not use a default queue manager under MSCS control. A queue manager does not have a property that makes it the default; IBM MQ keeps its own separate record. If you move a queue manager set to be the default to the other computer on failover, it does not become the default there. Make all your applications refer to specific queue managers by name.

Deleting a queue manager

Once a queue manager has moved node, its details exist in the registry on both computers. When you want to delete it, do so as normal on one computer, and then run the utility described in "IBM MQ MSCS support utility programs" on page 401 to clean up the registry on the other computer.

Support for existing queue managers

You can put an existing queue manager under MSCS control, provided that you can put your queue manager log files and queue files on a disk that is on the shared SCSI bus between the two machines (see Figure 73 on page 388). You need to take the queue manager offline briefly while the MSCS Resource is created.

If you want to create a new queue manager, create it independently of MSCS, test it, then put it under MSCS control. See:

- "Creating a queue manager for use with MSCS" on page 390
- "Moving a queue manager to MSCS storage" on page 391
- "Putting a queue manager under MSCS control" on page 392

Telling MSCS which queue managers to manage

You choose which queue managers are placed under MSCS control by using the MSCS Cluster Administrator to create a resource instance for each such queue manager. This process presents you with a list of resources from which to select the queue manager that you want that instance to manage.

Queue manager log files

When you move a queue manager to MSCS storage, you move its log and data files to a shared disk (for an example see "Moving a queue manager to MSCS storage" on page 391).

It is advisable before you move, to shut the queue manager cleanly and take a full backup of the data files and log files.

Multiple queue managers

IBM MQ MSCS support allows you to run multiple queue managers on each machine and to place individual queue managers under MSCS control.

Always use MSCS to manage clusters
Do not try to perform start and stop operations directly on any queue manager under the control of MSCS, using either the control commands or the IBM MQ Explorer. Instead, use MSCS Cluster Administrator to bring the queue manager online or take it offline.

Using the MSCS Cluster Administrator is partly to prevent possible confusion caused by MSCS reporting that the queue manager is offline, when in fact you have started it outside the control of MSCS. More seriously, stopping a queue manager without using MSCS is detected by MSCS as a failure, initiating failover to the other node.

Working in Active/Active mode
Both computers in the MSCS cluster can run queue managers in Active/Active mode. You do not need to have a completely idle machine acting as standby (but you can, if you want, in Active/Passive Mode).

If you plan to use both machines to run workload, provide each with sufficient capacity (processor, memory, secondary storage) to run the entire cluster workload at a satisfactory level of performance.

Note: If you are using MSCS together with Microsoft Transaction Server (COM+), you **cannot** use Active/Active mode. This is because, to use IBM MQ with MSCS and COM+:

- Application components that use IBM MQ COM+ support must run on the same computer as the Distributed Transaction Coordinator (DTC), a part of COM+.

- The queue manager must also run on the same computer.
- The DTC must be configured as an MSCS resource, and can therefore run on only one of the computers in the cluster at any time.

PostOnlineCommand and PreOfflineCommand
Use these commands to integrate IBM MQ MSCS support with other systems. You can use them to issue IBM MQ commands, wih some restrictions.

Specify these commands in the Parameters to a resource of type `IBM MQ MSCS`. You can use them to integrate IBM MQ MSCS support with other systems or procedures. For example, you could specify the name of a program that sends a mail message, activates a pager, or generates some other form of alert to be captured by another monitoring system.

PostOnlineCommand is invoked when the resource changes from offline to online; PreOfflineCommand is invoked for a change from online to offline. When invoked these commands are run, by default, from the Windows system directory. Because IBM MQ uses a 32-bit resource monitor process, on Windows 64-bit systems, this is the `\Windows\SysWOW64` directory rather than the `\Windows\system32` directory. For more information, see the Microsoft documentation about file redirection in a Windows x64 environment. Both commands run under the user account used to run the MSCS Cluster Service; and are invoked asynchronously; IBM MQ MSCS support does not wait for them to complete before continuing. This eliminates any risk that they might block or delay further cluster operations.

You can also use these commands to issue IBM MQ commands, for example to restart Requester channels. However, the commands are run at the point in time when the queue manager's state changes so they are not intended to perform long-running functions and must not make assumptions about the current state of the queue manager; it is quite possible that, immediately after the queue manager was brought online, an administrator issued an offline command.

If you want to run programs that depend on the state of the queue manager, consider creating instances of the `MSCS Generic Application` resource type, placing them in the same MSCS group as the queue manager resource, and making them dependent on the queue manager resource.

Using preferred nodes
It can be useful when using Active/Active mode to configure a *preferred node* for each queue manager. However, in general it is better not to set a preferred node but to rely on a manual failback.

Unlike some other relatively stateless resources, a queue manager can take a while to fail over (or back) from one node to the other. To avoid unnecessary outages, test the recovered node before failing a queue manager back to it. This precludes use of the `immediate` failback setting. You can configure failback to occur between certain times of day.

Probably the safest route is to move the queue manager back manually to the required node, when you are certain that the node is fully recovered. This precludes use of the `preferred node` option.

If COM+ errors occur in the Application Event log
When you install IBM MQ on a newly-installed MSCS cluster, you might find an error with Source COM+ and Event ID 4691 reported in the Application Event log.

This means that you are trying to run IBM MQ on a Microsoft Cluster Server (MSCS) environment when the Microsoft Distributed Transaction Coordinator (MSDTC) has not been configured to run in such an environment. For information on configuring MSDTC in a clustered environment, refer to Microsoft documentation.

IBM MQ MSCS support utility programs

A list of the IBM MQ support for MSCS utility programs that you can run at a command prompt.

IBM MQ support for MSCS includes the following utility programs:

Register/unregister the resource type
`haregtyp.exe`

After you *unregister* the IBM MQ MSCS resource type you can no longer create any resources of that type. MSCS does not let you unregister a resource type if you still have instances of that type within the cluster:

1. Using the MSCS Cluster Administrator, stop any queue managers that are running under MSCS control, by taking them offline as described in "Taking a queue manager offline from MSCS" on page 398.
2. Using the MSCS Cluster Administrator, delete the resource instances.
3. At a command prompt, unregister the resource type by entering the following command:

```
haregtyp /u
```

If you want to *register* the type (or re-register it at a later time), enter the following command at a command prompt:

```
haregtyp /r
```

After successfully registering the MSCS libraries, you must reboot the system if you have not done so since installing IBM MQ.

Move a queue manager to MSCS storage

`hamvmqm.exe`

See "Moving a queue manager to MSCS storage" on page 391.

Delete a queue manager from a node

`hadltmqm.exe`

Consider the case where you have had a queue manager in your cluster, it has been moved from one node to another, and now you want to destroy it. Use the IBM MQ Explorer to delete it on the node where it currently is. The registry entries for it still exist on the other computer. To delete these, enter the following command at a prompt on that computer:

```
hadltmqm /m qmname
```

where `qmname` is the name of the queue manager to remove.

Check and save setup details

`amqmsysn.exe`

This utility presents a dialog showing full details of your IBM MQ MSCS Support setup, such as might be requested if you call IBM support. There is an option to save the details to a file.

Multi-instance queue managers

Multi-instance queue managers are instances of the same queue manager configured on different servers. One instance of the queue manager is defined as the active instance and another instance is defined as the standby instance. If the active instance fails, the multi-instance queue manager restarts automatically on the standby server.

Example multi-instance queue manager configuration

Figure 74 on page 403 shows an example of a multi-instance configuration for queue manager QM1. IBM MQ is installed on two servers, one of which is a spare. One queue manager, QM1, has been created. One instance of QM1 is active, and is running on one server. The other instance of QM1 is running in standby on the other server, doing no active processing, but ready to take over from the active instance of QM1, if the active instance fails.

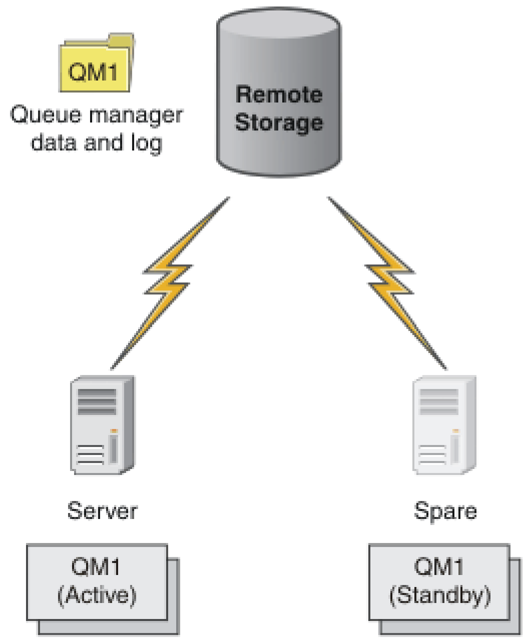

Figure 74. Multi-instance queue manager

When you intend to use a queue manager as a multi-instance queue manager, create a single queue manager on one of the servers using the **crtmqm** command, placing its queue manager data and logs in shared network storage. On the other server, rather than create the queue manager again, use the **addmqinf** command to create a reference to the queue manager data and logs on the network storage.

You can now run the queue manager from either of the servers. Each of the servers references the same queue manager data and logs; there is only one queue manager, and it is active on only one server at a time.

The queue manager can run either as a single instance queue manager, or as a multi-instance queue manager. In both cases only one instance of the queue manager is running, processing requests. The difference is that when running as a multi-instance queue manager, the server that is not running the active instance of the queue manager runs as a standby instance, ready to take over from the active instance automatically if the active server fails.

The only control you have over which instance becomes active first is the order in which you start the queue manager on the two servers. The first instance to acquire read/write locks to the queue manager data becomes the active instance.

You can swap the active instance to the other server, once it has started, by stopping the active instance using the switchover option to transfer control to the standby.

The active instance of QM1 has exclusive access to the shared queue manager data and logs folders when it is running. The standby instance of QM1 detects when the active instance has failed, and becomes the active instance. It takes over the QM1 data and logs in the state they were left by the active instance, and accepts reconnections from clients and channels.

The active instance might fail for various reasons that result in the standby taking over:

- Failure of the server hosting the active queue manager instance.
- Failure of connectivity between the server hosting the active queue manager instance and the file system.
- Unresponsiveness of queue manager processes, detected by IBM MQ, which then shuts down the queue manager.

You can add the queue manager configuration information to multiple servers, and choose any two servers to run as the active/standby pair. There is a limit of a total of two instances. You cannot have two standby instances and one active instance.

Additional components needed to build a high availability solution

A multi-instance queue manager is one part of a high availability solution. You need some additional components to build a useful high availability solution.

- Client and channel reconnection to transfer IBM MQ connections to the computer that takes over running the active queue manager instance.
- A high performance shared network file system (NFS) that manages locks correctly and provides protection against media and file server failure.

  Important: You must stop all multi-instance queue manager instances that are running in your environment before you can perform maintenance on the NFS drive. Make sure that you have queue manager configuration backups to recover, in the event of an NFS failure.
- Resilient networks and power supplies to eliminate single points of failure in the basic infrastructure.
- Applications that tolerate failover. In particular you need to pay close attention to the behavior of transactional applications, and to applications that browse IBM MQ queues.
- Monitoring and management of the active and standby instances to ensure that they are running, and to restart active instances that have failed. Although multi-instance queue managers restart automatically, you need to be sure that your standby instances are running, ready to take over, and that failed instances are brought back online as new standby instances.

IBM MQ MQI clients and channels reconnect automatically to the standby queue manager when it becomes active. More information about reconnection, and the other components in a high availability solution can be found in related topics. Automatic client reconnect is not supported by IBM MQ classes for Java.

Supported platforms

You can create a multi-instance queue manager on any of the non- z/OS platforms supported by IBM MQ from IBM WebSphere MQ 7.0.1.

Automatic client reconnection is supported for MQI clients from IBM WebSphere MQ 7.0.1 onwards.

Create a multi-instance queue manager

Create a multi-instance queue manager, creating the queue manager on one server, and configuring IBM MQ on another server. Multi-instance queue managers share queue manager data and logs.

Most of the effort involved in creating a multi-instance queue manager is the task of setting up the shared queue manager data and log files. You must create shared directories on network storage, and make the directories available to other servers using network shares. These tasks need to be performed by someone with administrative authority, such as *root* on UNIX and Linux systems. The steps are as follows:

1. Create the shares for the data and log files.
2. Create the queue manager on one server.
3. Run the command **dspmqinf** on the first server to collect the queue manager configuration data and copy it into the clipboard.
4. Run the command **addmqinf** with the copied data to create the queue manager configuration on the second server.

You do not run **crtmqm** to create the queue manager again on the second server.

File access control

You need to take care that the user and group `mqm` on all other servers have permission to access the shares.

On UNIX and Linux, you need to make the `uid` and `gid` of `mqm` the same on all the systems. You might need to edit `/etc/passwd` on each system to set a common `uid` and `gid` for `mqm`, and then reboot your system.

On Microsoft Windows, the user ID that is running the queue manager processes must have full control permission to the directories containing the queue manager data and log files. You can configure the permission in two ways:

1. Create a queue manager with a global group as the alternative security principal. Authorize the global group to have full control access to the directories containing queue manager data and log files; see “Secure shared queue manager data and log directories and files on Windows” on page 431. Make the user ID that is running the queue manager a member of the global group. You cannot make a local user a member of a global group, so the queue manager processes must run under a domain user ID. The domain user ID must be a member of the local group `mqm`. The task, “Create a multi-instance queue manager on domain workstations or servers” on page 407, demonstrates how to set up a multi-instance queue manager using files secured in this way.
2. Create a queue manager on the domain controller, so that the local `mqm` group has domain scope, "domain local". Secure the file share with the domain local `mqm`, and run queue manager processes on all instances of a queue manager under the same domain local `mqm` group. The task, “Create a multi-instance queue manager on domain controllers” on page 421, demonstrates how to set up a multi-instance queue manager using files secured in this way.

Configuration information

Configure as many queue manager instances as you need by modifying the IBM MQ queue manager configuration information about each server. Each server must have the same version of IBM MQ installed at a compatible fix level. The commands, **dspmqinf** and **addmqinf** assist you to configure the additional queue manager instances. Alternatively, you can edit the `mqs.ini` and `qm.ini` files directly. The topics, “Create a multi-instance queue manager on Linux” on page 443, “Create a multi-instance queue manager on domain workstations or servers” on page 407, and “Create a multi-instance queue manager on domain controllers” on page 421 are examples showing how to configure a multi-instance queue manager.

On Windows, UNIX and Linux systems, you can share a single `mqs.ini` file by placing it on the network share and setting the **AMQ_MQS_INI_LOCATION** environment variable to point to it.

Restrictions

1. Configure multiple instances of the same queue manager only on servers having the same operating system, architecture and endianness. For example, both machines must be either 32-bit or 64-bit.
2. All IBM MQ installations must be at release level 7.0.1 or higher.
3. Typically, active and standby installations are maintained at the same maintenance level. Consult the maintenance instructions for each upgrade to check whether you must upgrade all installations together.

   Note that the maintenance levels for the active and passive queue managers must be identical.
4. Share queue manager data and logs only between queue managers that are configured with the same IBM MQ user, group, and access control mechanism. IBM i For example, the network share set up on a Linux server could contain separate queue manager data and logs for UNIX and Linux queue managers, but could not contain the queue manager data used by IBM i.

   IBM i You can create multiple shares on the same networked storage for IBM i and other UNIX systems as long as the shares are different. You can give different shares different owners. The restriction is a consequence of the different names used for the IBM MQ users and groups between UNIX and IBM i. The fact that the user and group can have the same `uid` and `gid` does not relax the restriction.
5. On UNIX and Linux systems, configure the shared file system on networked storage with a `hard`, interruptible, mount rather than a `soft` mount. A hard interruptible mount forces the queue manager to hang until it is interrupted by a system call. Soft mounts do not guarantee data consistency after a server failure.

6. The shared log and data directories cannot be stored on a FAT, or an NFSv3 file system. For multi-instance queue managers on Windows, the networked storage must be accessed by the Common Internet File System (CIFS) protocol used by Windows networks.
7. z/OS z/OS does not support multi-instance queue managers. Use queue sharing groups.

   Reconnectable clients do work with z/OS queue managers.

Windows domains and multi-instance queue managers
A multi-instance queue manager on Windows requires its data and logs to be shared. The share must be accessible to all instances of the queue manager running on different servers or workstations. Configure the queue managers and share as part of a Windows domain. The queue manager can run on a domain workstation or server, or on the domain controller.

Before configuring a multi-instance queue manager, read "Secure unshared queue manager data and log directories and files on Windows" on page 434 and "Secure shared queue manager data and log directories and files on Windows" on page 431 to review how to control access to queue manager data and log files. The topics are educational; if you want to go directly to setting up shared directories for a multi-instance queue manager in a Windows domain; see "Create a multi-instance queue manager on domain workstations or servers" on page 407.

Run a multi-instance queue manager on domain workstations or servers

From Version 7.1, multi-instance queue managers run on a workstation or server that is a member of a domain. Before Version 7.1, multi-instance queue managers ran only on domain controllers; see "Run a multi-instance queue manager on domain controllers" on page 407. To run a multi-instance queue manager on Windows, you require a domain controller, a file server, and two workstations or servers running the same queue manager connected to the same domain.

The change that makes it possible to run a multi-instance queue manager on any server or workstation in a domain, is that you can now create a queue manager with an additional security group. The additional security group is passed in the **crtmqm** command, in the `-a` parameter. You secure the directories that contain the queue manager data and logs with the group. The user ID that runs queue manager processes must be a member of this group. When the queue manager accesses the directories, Windows checks the permissions the user ID has to access the directories. By giving both the group and the user ID domain scope, the user ID running the queue manager processes has credentials from the global group. When the queue manager is running on a different server, the user ID running the queue manager processes can have the same credentials. The user ID does not have to be the same. It has to be a member of the alternative security group, as well as a member of the local `mqm` group.

The task of creating a multi-instance queue manager is the same as in Version 7.0.1 with one change. You must add the additional security group name to the parameters of the **crtmqm** command. The task is described in "Create a multi-instance queue manager on domain workstations or servers" on page 407.

Multiple steps are required to configure the domain, and the domain servers and workstations. You must understand how Windows authorizes access by a queue manager to its data and log directories. If you are not sure how queue manager processes are authorized to access their log and data files read the topic "Secure unshared queue manager data and log directories and files on Windows" on page 434. The topic includes two tasks to help you understand the steps the required. The tasks are "Reading and writing data and log files authorized by the local mqm group" on page 436 and "Reading and writing data and log files authorized by an alternative local security group" on page 439. Another topic, "Secure shared queue manager data and log directories and files on Windows" on page 431, explains how to secure shared directories containing queue manager data and log files with the alternative security group. The topic includes four tasks, to set up a Windows domain, create a file share, install IBM MQ for Windows, and configure a queue manager to use the share. The tasks are as follows:

1. "Creating an Active Directory and DNS domain for IBM MQ" on page 410.
2. "Installing IBM MQ on a server or workstation in a Windows domain" on page 413.
3. "Creating a shared directory for queue manager data and log files" on page 416.

4. “Reading and writing shared data and log files authorized by an alternative global security group” on page 418.

You can then do the task, “Create a multi-instance queue manager on domain workstations or servers” on page 407, using the domain. Do these tasks to explore setting up a multi-instance queue manager before transferring your knowledge to a production domain.

Run a multi-instance queue manager on domain controllers

In Version 7.0.1, multi-instance queue managers ran only on domain controllers. Queue manager data could be secured with the domain `mqm` group. As the topic “Secure shared queue manager data and log directories and files on Windows” on page 431 explains, you cannot share directories secured with the local `mqm` group on workstations or servers. However on domain controllers all group and principals have domain scope. If you install IBM MQ for Windows on a domain controller, the queue manager data and log files are secured with the domain `mqm` group, which can be shared. Follow the steps in the task, “Create a multi-instance queue manager on domain controllers” on page 421 to configure a multi-instance queue manager on domain controllers.

Related information

Managing Authorization and Access Control

How to use Windows Server cluster nodes as domain controllers

Create a multi-instance queue manager on domain workstations or servers
An example shows how to set up a multi-instance queue manager on Windows on a workstation or a server that is part of a Windows domain. The server does not have to be a domain controller. The setup demonstrates the concepts involved, rather than being production scale. The example is based on Windows Server 2008. The steps might differ on other versions of Windows Server.

In a production scale configuration, you might have to tailor the configuration to an existing domain. For example, you might define different domain groups to authorize different shares, and to group the user IDs that run queue managers.

The example configuration consists of three servers:

sun
: A Windows Server 2008 domain controller. It owns the *wmq.example.com* domain that contains *Sun* , *mars* , and *venus* . For the purposes of illustration, it is also used as the file server.

mars
: A Windows Server 2008 used as the first IBM MQ server. It contains one instance of the multi-instance queue manager called *QMGR* .

venus
: A Windows Server 2008 used as the second IBM MQ server. It contains the second instance of the multi-instance queue manager called *QMGR* .

Replace the italicized names in the example, with names of your choosing.

Before you begin

On Windows, you do not need to verify the file system that you plan to store queue manager data and log files on. The checking procedure, Verifying shared file system behavior, is applicable to UNIX and Linux. On Windows, the checks are always successful.

Do the steps in the following tasks. The tasks create the domain controller and domain, install IBM MQ for Windows on one server, and create the file share for data and log files. If you are configuring an existing domain controller, you might find it useful to try out the steps on a new Windows Server 2008. You can adapt the steps to your domain.

1. “Creating an Active Directory and DNS domain for IBM MQ” on page 410.
2. “Installing IBM MQ on a server or workstation in a Windows domain” on page 413.
3. “Creating a shared directory for queue manager data and log files” on page 416.

4. “Reading and writing shared data and log files authorized by an alternative global security group” on page 418.

About this task

This task is one of a sequence of tasks to configure a domain controller and two servers in the domain to run instances of a queue manager. In this task you configure a second server, *venus* , to run another instance of the queue manager *QMGR* . Follow the steps in this task to create the second instance of the queue manager, *QMGR*, and test that it works.

This task is separate from the four tasks in the preceding section. It contains the steps that convert a single instance queue manager into a multi-instance queue manager. All the other steps are common to single or multi-instance queue managers.

Procedure

1. Configure a second server to run IBM MQ for Windows.
   a) Do the steps in the task “Installing IBM MQ on a server or workstation in a Windows domain” on page 413 to create a second domain server. In this sequence of tasks the second server is called *venus* .

      Tip: Create the second installation using the same installation defaults for IBM MQ on each of the two servers. If the defaults differ, you might have to tailor the `Prefix` and the `InstallationName` variables in the *QMGR* **QueueManager** stanza in the IBM MQ configuration file `mqs.ini`. The variables refer to paths that can differ for each installation and queue manager on each server. If the paths remain the same on every server, it is simpler to configure a multi-instance queue manager.
2. Create a second instance of *QMGR* on *venus* .
   a) If *QMGR* on *mars* does not exist, do the task “Reading and writing shared data and log files authorized by an alternative global security group” on page 418, to create it
   b) Check the values of the `Prefix` and `InstallationName` parameters are correct for *venus* .

      On *mars* , run the **dspmqinf** command:

      ```
      dspmqinf QMGR
      ```

      The system response:

      ```
      QueueManager:
      Name=QMGR
      Directory=QMGR
      Prefix=C:\ProgramData\IBM\MQ
      DataPath=\\sun\wmq\data\QMGR
      InstallationName=Installation1
      ```
   c) Copy the machine-readable form of the **QueueManager** stanza to the clipboard.

      On *mars* run the **dspmqinf** command again, with the `-o command` parameter.

      ```
      dspmqinf -o command QMGR
      ```

      The system response:

      ```
      addmqinf -s QueueManager -v Name=QMGR
      -v Directory=QMGR -v Prefix="C:\ProgramData\IBM\MQ"
      -v DataPath=\\sun\wmq\data\QMGR
      ```

d) On *venus* run the **addmqinf** command from the clipboard to create an instance of the queue manager on *venus* .

Adjust the command, if necessary, to accommodate differences in the `Prefix` or `InstallationName` parameters.

```
addmqinf -s QueueManager -v Name=QMGR
-v Directory=QMGR -v Prefix="C:\ProgramData\IBM\MQ"
-v DataPath=\\sun\wmq\data\QMGR
```

```
IBM MQ configuration information added.
```

3. Start the queue manager *QMGR* on *venus* , permitting standby instances.

a) Check *QMGR* on *mars* is stopped.

On *mars* , run the **dspmq** command:

```
dspmq -m QMGR
```

The system response depends on how the queue manager was stopped; for example:

```
C:\Users\Administrator>dspmq -m QMGR
QMNAME(QMGR) STATUS(Ended immediately)
```

b) On *venus* run the **strmqm** command to start *QMGR* permitting standbys:

```
strmqm -x QMGR
```

The system response:

```
IBM MQ queue manager 'QMGR' starting.
The queue manager is associated with installation 'Installation1'.
5 log records accessed on queue manager 'QMGR' during the log
replay phase.
Log replay for queue manager 'QMGR' complete.
Transaction manager state recovered for queue manager 'QMGR'.
IBM MQ queue manager 'QMGR' started using V7.1.0.0.
```

Results

To test the multi-instance queue manager switches over, do the following steps:

1. On *mars* , run the **strmqm** command to start *QMGR* permitting standbys:

```
strmqm -x QMGR
```

The system response:

```
IBM MQ queue manager 'QMGR' starting.
The queue manager is associated with installation 'Installation1'.
A standby instance of queue manager 'QMGR' has been started.
The active instance is running elsewhere.
```

2. On *venus* run the **endmqm** command:

```
endmqm -r -s -i QMGR
```

The system response on *venus* :

```
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ended, permitting switchover to
a standby instance.
```

And on *mars* :

```
dspmq
QMNAME(QMGR) STATUS(Running as standby)
C:\Users\wmquser2>dspmq
QMNAME(QMGR) STATUS(Running as standby)
C:\Users\wmquser2>dspmq
QMNAME(QMGR)  STATUS(Running)
```

What to do next

To verify a multi-instance queue manager using sample programs; see "Verify the multi-instance queue manager on Windows" on page 428.

Creating an Active Directory and DNS domain for IBM MQ

This task creates the domain *wmq.example.com* on a Windows 2008 domain controller called *sun* . It configures the Domain mqm global group in the domain, with the correct rights, and with one user.

In a production scale configuration, you might have to tailor the configuration to an existing domain. For example, you might define different domain groups to authorize different shares, and to group the user IDs that run queue managers.

The example configuration consists of three servers:

sun
A Windows Server 2008 domain controller. It owns the *wmq.example.com* domain that contains *Sun* , *mars* , and *venus* . For the purposes of illustration, it is also used as the file server.

mars
A Windows Server 2008 used as the first IBM MQ server. It contains one instance of the multi-instance queue manager called *QMGR* .

venus
A Windows Server 2008 used as the second IBM MQ server. It contains the second instance of the multi-instance queue manager called *QMGR* .

Replace the italicized names in the example, with names of your choosing.

Before you begin

1. The task steps are consistent with a Windows Server 2008 that is installed but not configured with any roles. If you are configuring an existing domain controller, you might find it useful to try out the steps on a new Windows Server 2008. You can adapt the steps to your domain.

About this task

In this task, you create an Active Directory and DNS domain on a new domain controller. You then configure it ready to install IBM MQ on other servers and workstations that join the domain. Follow the task if you are unfamiliar with installing and configuring Active Directory to create a Windows domain. You must create a Windows domain in order to create a multi-instance queue manager configuration. The task is not intended to guide you in the best way to configure a Windows domain. To deploy multi-instance queue managers in a production environment, you must consult Windows documentation.

During the task you do the following steps:

1. Install Active Directory.
2. Add a domain.
3. Add the domain to DNS.
4. Create the global group `Domain mqm` and give it the correct rights.
5. Add a user and make it a member of the global group `Domain mqm`.

This task is one of a set of related tasks that illustrate accessing queue manager data and log files. The tasks show how to create a queue manager authorized to read and write data and log files that are stored in a directory of your choosing. They accompany the task, "Windows domains and multi-instance queue managers" on page 406.

For the purposes of the task the domain controller hostname is *sun* , and the two IBM MQ servers are called *mars* and *venus* . The domain is called *wmq.example.com* . You can replace all the italicized names in the task with names of your own choosing.

Procedure

1. Log on to the domain controller, *sun* , as the local or `Workgroup` administrator.

   If the server is already configured as a domain controller, you must log on as a domain administrator.
2. Run the Active Directory Domain Services wizard.
   a) Click **Start** > **Run...** Type `dcpromo` and click **OK**.

   If the Active Directory binary files are not already installed, Windows installs the files automatically.
3. In the first window of the wizard, leave the **Use advanced mode installation** check box clear. Click **Next** > **Next** and click **Create a new domain in a new forest** > **Next**.
4. Type *wmq.example.com* into the **FQDN of the forest root domain** field. Click **Next**.
5. In the Set Forest Functional Level window, select **Windows Server 2003**, or later, from the list of **Forest functional levels** > **Next**.

   The oldest level of Windows Server that is supported by IBM MQ is Windows Server 2003.
6. Optional: In the Set Domain Functional Level window, select **Windows Server 2003**, or later, from the list of **Domain functional levels** > **Next**.

   This step is only required if you set the Forest Functional Level to **Windows Server 2003**.
7. The Additional Domain Controller Options window opens, with **DNS server** selected as an additional option. Click **Next** and **Yes** to clear the warning window.

   Tip: If a DNS server is already installed this option is not presented to you. If you want to follow this task precisely, remove all the roles from this domain controller and start again.
8. Leave the `Database`, `Log Files`, and SYSVOL directories unchanged; click **Next**.
9. Type a password into the **Password** and **Confirm password** fields in the Directory Services Restore Mode Administrator Password window. Click **Next** > **Next**. Select **Reboot on completion** in the final wizard window.
10. When the domain controller reboots, log on as *wmq*`\Adminstrator`.

    The server manager starts automatically.
11. Open the *wmq.example.com*`\Users` folder
    a) Open **Server Manager** > **Roles** > **Active Directory Domain Services** > ***wmq.example.com*** > **Users**.
12. Right-click **Users** > **New** > **Group**.
    a) Type a group name into the **Group name** field.

Note: The preferred group name is `Domain mqm`. Type it exactly as shown.

- Calling the group `Domain mqm` modifies the behavior of the "Prepare IBM MQ " wizard on a domain workstation or server. It causes the "Prepare IBM MQ " wizard automatically to add the group `Domain mqm` to the local `mqm` group on each new installation of IBM MQ in the domain.
- You can install workstations or servers in a domain with no `Domain mqm` global group. If you do so, you must define a group with the same properties as `Domain mqm` group. You must make that group, or the users that are members of it, members of the local `mqm` group wherever IBM MQ is installed in a domain. You can place domain users into multiple groups. Create multiple domain groups, each group corresponding to a set of installations that you want to manage separately. Split domain users, according to the installations they manage, into different domain groups. Add each domain group or groups to the local `mqm` group of different IBM MQ installations. Only domain users in the domain groups that are members of a specific local `mqm` group can create, administer, and run queue managers for that installation.
- The domain user that you nominate when installing IBM MQ on a workstation or server in a domain must be a member of the `Domain mqm` group, or of an alternative group you defined with same properties as the `Domain mqm` group.

b) Leave **Global** clicked as the **Group scope**, or change it to **Universal**. Leave **Security** clicked as the **Group type**. Click **OK**.

13. Add the rights, **Allow Read group membership** and **Allow Read groupMembershipSAM** to the rights of the `Domain mqm` global group.

a) In the Server Manager action bar, click **View** > **Advanced features**

b) In the Server Manager navigation tree, click **Users**

c) In the Users window, right-click **Domain mqm** > **Properties**

d) Click **Security** > **Advanced** > **Add...**. Type `Domain mqm` and click **Check names** > **OK**.

The **Name** field is prefilled with the string, `Domain mqm (`*`domain name`*`\Domain mqm)`.

e) Click **Properties**. In the **Apply to** list, select **Descendant User Objects**.

f) From the **Permissions** list, select the **Read group membership** and **Read groupMembershipSAM** **Allow** check boxes; click **OK** > **Apply** > **OK** > **OK**.

14. Add two or more users to the `Domain mqm` global group.

One user, *`wmquser1`* in the example, runs the IBM MQ service, and the other user, *`wmquser2`* , is used interactively.

A domain user is required to create a queue manager that uses the alternative security group in a domain configuration. It is not sufficient for the user ID to be an administrator, although an administrator has authority to run the **`crtmqm`** command. The domain user, who could be an administrator, must be a member of the local `mqm` group as well as of the alternative security group.

In the example, you make *`wmquser1`* and *`wmquser2`* members of the `Domain mqm` global group. The "Prepare IBM MQ " wizard automatically configures `Domain mqm` as a member of the local `mqm` group where ever the wizard is run.

You must provide a different user to run the IBM MQ service for each installation of IBM MQ on a single computer. You can reuse the same users on different computers.

a) In the Server Manager navigation tree, click **Users** > **New** > **User**

b) In the New Object - User window, type *`wmquser1`* into the **User logon name** field. Type *`WebSphere`* into the **First name** field, and *`MQ1`* into the **Last name** field. Click **Next**.

c) Type a password into the **Password** and **Confirm password** fields, and clear the **User must change password at next logon** check box. Click **Next** > **Finish**.

d) In the Users window, right-click ***WebSphere MQ*** > **Add to a group...**. Type `Domain mqm` and click **Check Names** > **OK** > **OK**.

e) Repeat steps a to d to add *`WebSphere MQ2`* as *`wmquser2`* .

15. Running IBM MQ as a service.

If you need to run IBM MQ as a service, and then give the domain user (that you obtained from your domain administrator) the access to run as a service, carry out the following procedure:

a) Click **Start > Run...**.

   Type the command `secpol.msc` and click **OK**.

b) Open **Security Settings > Local Policies > User Rights Assignments**.

   In the list of policies, right-click **Log on as a service > Properties**.

c) Click **Add User or Group...**

   Type the name of the user you obtained from your domain administrator, and click **Check Names**

d) If prompted by a Windows Security window, type the user name and password of an account user or administrator with sufficient authority, and click **OK > Apply > OK**.

   Close the Local Security Policy window.

Note: On Windows Server 2008 and Windows Server 2012 the User Account Control (UAC) is enabled by default.

The UAC feature restricts the actions users can perform on certain operating system facilities, even if they are members of the Administrators group. You must take appropriate steps to overcome this restriction.

What to do next

Proceed to the next task, "Installing IBM MQ on a server or workstation in a Windows domain" on page 413.

Related tasks

Installing IBM MQ on a server or workstation in a Windows domain

Creating a shared directory for queue manager data and log files

Reading and writing shared data and log files authorized by an alternative global security group

Installing IBM MQ on a server or workstation in a Windows domain

In this task, you install and configure IBM MQ on a server or workstation in the *wmq.example.com* Windows domain.

In a production scale configuration, you might have to tailor the configuration to an existing domain. For example, you might define different domain groups to authorize different shares, and to group the user IDs that run queue managers.

The example configuration consists of three servers:

sun
: A Windows Server 2008 domain controller. It owns the *wmq.example.com* domain that contains *Sun* , *mars* , and *venus* . For the purposes of illustration, it is also used as the file server.

mars
: A Windows Server 2008 used as the first IBM MQ server. It contains one instance of the multi-instance queue manager called *QMGR* .

venus
: A Windows Server 2008 used as the second IBM MQ server. It contains the second instance of the multi-instance queue manager called *QMGR* .

Replace the italicized names in the example, with names of your choosing.

Before you begin

1. Do the steps in "Creating an Active Directory and DNS domain for IBM MQ" on page 410 to create a domain controller, *sun* , for the domain *wmq.example.com* . Change the italicized names to suit your configuration.
2. See Hardware and software requirements on Windows systems for other Windows versions you can run IBM MQ on.

About this task

In this task you configure a Windows Server 2008, called *mars* , as a member of the *wmq.example.com* domain. You install IBM MQ, and configure the installation to run as a member of the *wmq.example.com* domain.

This task is one of a set of related tasks that illustrate accessing queue manager data and log files. The tasks show how to create a queue manager authorized to read and write data and log files that are stored in a directory of your choosing. They accompany the task, “Windows domains and multi-instance queue managers” on page 406.

For the purposes of the task the domain controller hostname is *sun* , and the two IBM MQ servers are called *mars* and *venus* . The domain is called *wmq.example.com* . You can replace all the italicized names in the task with names of your own choosing.

Procedure

1. Add the domain controller, *sun.wmq.example.com* to *mars* as a DNS server.
   a) On *mars* , log on as *mars*`\Administrator` and click **Start**.
   b) Right-click **Network** > **Properties** > **Manage network connections**.
   c) Right-click the network adapter, click **Properties**.

      The system responds with the Local Area Connection Properties window listing items the connection uses.
   d) Select the **Internet Protocol Version 4** or **Internet Protocol Version 6** from the list of items in the Local Area Connection Properties window. Click **Properties** > **Advanced...** and click the **DNS** tab.
   e) Under the DNS server addresses, click **Add...**.
   f) Type the IP address of the domain controller, which is also the DNS server, and click **Add**.
   g) Click **Append these DNS suffixes** > **Add...**.
   h) Type *wmq.example.com* and click **Add**.
   i) Type *wmq.example.com* in the **DNS suffix for this connection** field.
   j) Select **Register this connection's address in DNS** and **Use this connection's suffix in DNS registration**. Click **OK** > **OK** > **Close**
   k) Open a command window, and type the command **`ipconfig /all`** to review the TCP/IP settings.
2. On *mars* , add the computer to the *wmq.example.com* domain.
   a) Click **Start**
   b) Right-click **Computer** > **Properties**. In the Computer name, domain and workgroup settings division, click **Change settings**.
   c) In the System Properties windows, click **Change...**.
   d) Click Domain, type *wmq.example.com* , and click **OK**.
   e) Type the **User name** and **Password** of the domain controller administrator, who has the authority to permit the computer to join the domain, and click **OK**.
   f) Click **OK** > **OK** > **Close** > **Restart Now** in response to the "Welcome to the *wmq.example.com* domain" message.
3. Check that the computer is a member of the *wmq.example.com* domain
   a) On *sun* , log on to the domain controller as *wmq*`\Administrator`.
   b) Open **Server Manager** > **Active Directory Domain Services** > ***wmq.example.com*** > **Computers** and check *mars* is listed correctly in the Computers window.
4. Install IBM MQ for Windows on *mars* .

   For further information about running the IBM MQ for Windows installation wizard; see Installing IBM MQ server on Windows .

a) On *mars* , log on as the local administrator, *mars*\Administrator.

b) Run the **Setup** command on the IBM MQ for Windows installation media.

   The IBM MQ Launchpad application starts.

c) Click **Software Requirements** to check that the prerequisite software is installed.

d) Click **Network Configuration** > **Yes** to configure a domain user ID.

   The task, "Creating an Active Directory and DNS domain for IBM MQ" on page 410, configures a domain user ID for this set of tasks.

e) Click **IBM MQ Installation**, select an installation language and click Launch IBM MQ Installer.

f) Confirm the license agreement and click **Next** > **Next** > **Install** to accept the default configuration. Wait for the installation to complete, and click **Finish**.

   You might want to change the name of the installation, install different components, configure a different directory for queue manager data and logs, or install into a different directory. If so, click **Custom** rather than **Typical**.

   IBM MQ is installed, and the installer starts the "Prepare IBM MQ " wizard.

   Important: Do not run the wizard yet.

5. Configure the user that is going to run the IBM MQ service with the **Run as a service** right.

   Choose whether to configure the local mqm group, the Domain mqm group, or the user that is going to run the IBM MQ service with the right. In the example, you give the user the right.

   a) Click **Start** > **Run...**, type the command **secpol.msc** and click **OK**.

   b) Open **Security Settings** > **Local Policies** > **User Rights Assignments**. In the list of policies, right-click **Log on as a service** > **Properties** .

   c) Click **Add User or Group...** and type *wmquser1* and click **Check Names**

   d) Type the user name and password of a domain administrator, *wmq*\Administrator, and click **OK** > **Apply** > **OK**. Close the Local Security Policy window.

6. Run the "Prepare IBM MQ " wizard.

   For further information about running the "Prepare IBM MQ " wizard; see Configuring IBM MQ with the Prepare IBM MQ wizard.

   a) The IBM MQ Installer runs the "Prepare IBM MQ " automatically.

      To start the wizard manually, find the shortcut to the "Prepare IBM MQ " in the **Start** > **All programs** > **IBM MQ** folder. Select the shortcut that corresponds to the installation of IBM MQ in a multi-installation configuration.

   b) Click **Next** and leave **Yes** clicked in response to the question "Identify if there is a Windows 2000 or later domain controller in the network".

   c) Click **Yes** > **Next** in the first Configuring IBM MQ for Windows for Windows domain users window.

   d) In the second Configuring IBM MQ for Windows for Windows domain users window, type *wmq* in the **Domain** field. Type *wmquser1* in the **User name** field, and the password, if you set one, in the **Password** field. Click **Next**.

      The wizard configures and starts the IBM MQ with *wmquser1*.

   e) In the final page of the wizard, select or clear the check boxes as you require and click **Finish**.

What to do next

1. Do the task, "Reading and writing data and log files authorized by the local mqm group" on page 436, to verify that the installation and configuration are working correctly.
2. Do the task, "Creating a shared directory for queue manager data and log files" on page 416, to configure a file share to store the data and log files of a multi-instance queue manager.

Related concepts

User rights required for an IBM MQ Windows Service

Related tasks

Creating an Active Directory and DNS domain for IBM MQ

Creating a shared directory for queue manager data and log files

Reading and writing shared data and log files authorized by an alternative global security group

Creating a shared directory for queue manager data and log files

This task is one of a set of related tasks that illustrate accessing queue manager data and log files. The tasks show how to create a queue manager authorized to read and write data and log files that are stored in a directory of your choosing.

In a production scale configuration, you might have to tailor the configuration to an existing domain. For example, you might define different domain groups to authorize different shares, and to group the user IDs that run queue managers.

The example configuration consists of three servers:

sun
: A Windows Server 2008 domain controller. It owns the *wmq.example.com* domain that contains *Sun* , *mars* , and *venus* . For the purposes of illustration, it is also used as the file server.

mars
: A Windows Server 2008 used as the first IBM MQ server. It contains one instance of the multi-instance queue manager called *QMGR* .

venus
: A Windows Server 2008 used as the second IBM MQ server. It contains the second instance of the multi-instance queue manager called *QMGR* .

Replace the italicized names in the example, with names of your choosing.

Before you begin

1. To do this task exactly as documented, do the steps in the task, "Creating an Active Directory and DNS domain for IBM MQ" on page 410, to create the domain *sun.wmq.example.com* on the domain controller *sun* . Change the italicized names to suit your configuration.

About this task

This task is one of a set of related tasks that illustrate accessing queue manager data and log files. The tasks show how to create a queue manager authorized to read and write data and log files that are stored in a directory of your choosing. They accompany the task, "Windows domains and multi-instance queue managers" on page 406.

In the task, you create a share containing a data and log directory, and a global group to authorize access to the share. You pass the name of the global group that authorizes the share to the **crtmqm** command in its -a parameter. The global group gives you the flexibility of separating the users of this share from users of other shares. If you do not need this flexibility, authorize the share with the Domain mqm group rather than create a new global group.

The global group used for sharing in this task is called *wmqha* , and the share is called *wmq* . They are defined on the domain controller *sun* in the Windows domain *wmq.example.com* . The share has full control permissions for the global group *wmqha* . Replace the italicized names in the task with names of your choosing.

For the purposes of this task the domain controller is the same server as the file server. In practical applications, split the directory and file services between different servers for performance and availability.

You must configure the user ID that the queue manager is running under to be a member of two groups. It must be a member of the local mqm group on an IBM MQ server, and of the *wmqha* global group.

In this set of tasks, when the queue manager is running as a service, it runs under the user ID *wmquser1* , so *wmquser1* must be a member of *wmqha* . When the queue manager is running interactively, it runs under the user ID *wmquser2* , so *wmquser2* must be a member of *wmqha* . Both *wmquser1* and *wmquser2* are members of the global group `Domain mqm`. `Domain mqm` is a member of the local `mqm` group on the *mars* and *venus* IBM MQ servers. Hence, *wmquser1* and *wmquser2* are members of the local `mqm` group on both IBM MQ servers.

Procedure

1. Log on to the domain controller, *sun.wmq.example.com* as the domain administrator.
2. Create the global group *wmqha* .
    a) Open **Server Manager** > **Roles** > **Active Directory Domain Services** > ***wmq.example.com*** > **Users**.
    b) Open the *wmq.example.com*`\Users` folder
    c) Right-click **Users** > **New** > **Group**.
    d) Type *wmqha* into the **Group name** field.
    e) Leave **Global** clicked as the **Group scope** and **Security** as the **Group type**. Click **OK**.
3. Add the domain users *wmquser1* and *wmquser2* to the global group, *wmqha* .
    a) In the Server Manager navigation tree, click **Users** and right-click ***wmqha*** > **Properties** in the list of users.
    b) Click the Members tab in the *wmqha* Properties window.
    c) Click **Add...** ; type *wmquser1 ; wmquser2* and click **Check Names** > **OK** > **Apply** > **OK**.
4. Create the directory tree to contain queue manager data and log files.
    a) Open a command prompt.
    b) Type the command:

```
md c:\wmq\data, c:\wmq\logs
```

5. Authorize the global group *wmqha* to have full control permission to the *c:\wmq* directories and share.
    a) In Windows Explorer, right-click ***c:\wmq*** > **Properties**.
    b) Click the **Security** tab and click **Advanced** > **Edit...**.
    c) Clear the check box for **Include inheritable permissions from this object's owner**. Click **Copy** in the Windows Security window.
    d) Select the lines for Users in the list of **Permission entries** and click **Remove**. Leave the lines for SYSTEM, Administrators, and CREATOR OWNER in the list of **Permission entries**.
    e) Click **Add...**, and type the name of the global group *wmqha* . Click **Check Names** > **OK**.
    f) In the Permission Entry for `wmq` window, select **Full Control** in the list of **Permissions**.
    g) Click **OK** > **Apply** > **OK** > **OK** > **OK**
    h) In Windows Explorer, right-click ***c:\wmq*** > **Share...**.
    i) Click **Advanced Sharing...** and select the **Share this folder** check box. Leave the share name as *wmq* .
    j) Click **Permissions** > **Add...**, and type the name of the global group *wmqha* . Click **Check Names** > **OK**.
    k) Select *wmqha* in the list of **Group or user names**. Select the **Full Control** check box in the list of **Permissions for *wmqha*** ; click **Apply**.
    l) Select *Administrators* in the list of **Group or user names**. Select the **Full Control** check box in the list of **Permissions for *Administrators*** ; click **Apply** > **OK** > **OK** > **Close**.

What to do next

Check that you can read and write files to the shared directories from each of the IBM MQ servers. Check the IBM MQ service user ID, *wmquser1* and the interactive user ID, *wmquser2*.

1. If you are using remote desktop, you must add *wmq**wmquser1* and *wmquser2* to the local group `Remote Desktop Users` on *mars* .
    a. Log on to *mars* as *wmq*\\`Administrator`
    b. Run the **`lusrmgr.msc`** command to open the Local Users and Groups window.
    c. Click **Groups**. Right-click **Remote Desktop Users** > **Properties** > **Add...**. Type *wmquser1* ; *wmquser2* and click **Check Names**.
    d. Type in the user name and password of the domain administrator, *wmq*\\`Administrator`, and click **OK** > **Apply** > **OK**.
    e. Close the Local Users and Groups window.
2. Log on to *mars* as *wmq**wmquser1*.
    a. Open a Windows Explorer window, and type in *sun**wmq* .

       The system responds by opening the *wmq* share on *sun.wmq.example.com* , and lists the data and logs directories.
    b. Check the permissions of *wmquser1* by creating a file in data subdirectory, adding some content, reading it, and then deleting it.
3. Log on to *mars* as *wmq**wmquser2*, and repeat the checks.
4. Do the next task, to create a queue manager to use the shared data and log directories; see "Reading and writing shared data and log files authorized by an alternative global security group" on page 418.

Related tasks

Creating an Active Directory and DNS domain for IBM MQ

Installing IBM MQ on a server or workstation in a Windows domain

Reading and writing shared data and log files authorized by an alternative global security group

Reading and writing shared data and log files authorized by an alternative global security group
This task shows how to use the `-a` flag on the **`crtmqm`** command. The `-a` flag gives the queue manager access to its log and data files on a remote file share using the alternative security group.

In a production scale configuration, you might have to tailor the configuration to an existing domain. For example, you might define different domain groups to authorize different shares, and to group the user IDs that run queue managers.

The example configuration consists of three servers:

sun
A Windows Server 2008 domain controller. It owns the *wmq.example.com* domain that contains *Sun* , *mars* , and *venus* . For the purposes of illustration, it is also used as the file server.

mars
A Windows Server 2008 used as the first IBM MQ server. It contains one instance of the multi-instance queue manager called *QMGR* .

venus
A Windows Server 2008 used as the second IBM MQ server. It contains the second instance of the multi-instance queue manager called *QMGR* .

Replace the italicized names in the example, with names of your choosing.

Before you begin

Do the steps in the following tasks. The tasks create the domain controller and domain, install IBM MQ for Windows on one server, and create the file share for data and log files. If you are configuring an existing domain controller, you might find it useful to try out the steps on a new Windows Server 2008. You can adapt the steps to your domain.

1. "Creating an Active Directory and DNS domain for IBM MQ" on page 410.
2. "Installing IBM MQ on a server or workstation in a Windows domain" on page 413.
3. "Creating a shared directory for queue manager data and log files" on page 416.

About this task

This task is one of a set of related tasks that illustrate accessing queue manager data and log files. The tasks show how to create a queue manager authorized to read and write data and log files that are stored in a directory of your choosing. They accompany the task, "Windows domains and multi-instance queue managers" on page 406.

In this task, you create a queue manager that stores its data and logs in a remote directory on a file server. For the purposes of this example, the file server is the same server as the domain controller. The directory containing the data and log folders is shared with full control permission given to the global group wmqha.

Procedure

1. Log on to the domain server, *mars* , as the local administrator, *mars*`\Administrator`.
2. Open a command window.
3. Restart the IBM MQ service.

   You must restart the service so that the user ID it runs under acquires the additional security credentials you configured for it.

   Type the commands:

   ```
   endmqsvc
   strmqsvc
   ```

   The system responses:

   ```
   5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
   The MQ service for installation 'Installation1' ended successfully.
   ```

   And:

   ```
   5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
   The MQ service for installation 'Installation1' started successfully.
   ```
4. Create the queue manager.

   ```
   crtmqm -a wmq\wmqha -sax -u SYSTEM.DEAD.LETTER.QUEUE -md \\sun\wmq\data -ld \\sun\wmq\logs
   QMGR
   ```

   You must specify the domain, *wmq* , of the alternative security group *wmqha* by specifying full domain name of the global group *"wmq\wmqha"* .

   You must spell out the Universal Naming Convention (UNC) name of the share *\\sun\wmq* , and not use a mapped drive reference.

   The system response:

   ```
   IBM MQ queue manager created.
   Directory '\\sun\wmq\data\QMGR' created.
   The queue manager is associated with installation '1'
   Creating or replacing default objects for queue manager 'QMGR'
   ```

```
Default objects statistics : 74 created. 0 replaced.
Completing setup.
Setup completed.
```

What to do next

Test the queue manager by putting and getting a message to a queue.

1. Start the queue manager.

   ```
   strmqm QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager 'QMGR' starting.
   The queue manager is associated with installation '1'.
   5 log records accessed on queue manager 'QMGR' during the log
   replay phase.
   Log replay for queue manager 'QMGR' complete.
   Transaction manager state recovered for queue manager 'QMGR'.
   IBM MQ queue manager 'QMGR' started using V7.1.0.0.
   ```

2. Create a test queue.

   ```
   echo define qlocal(QTEST) | runmqsc QMGR
   ```

   The system response:

   ```
   5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
   Starting MQSC for queue manager QMGR.


   1 : define qlocal(QTEST)
   AMQ8006: IBM MQ queue created.
   One MQSC command read.
   No commands have a syntax error.
   All valid MQSC commands were processed.
   ```

3. Put a test message using the sample program **amqsput**.

   ```
   echo 'A test message' | amqsput QTEST QMGR
   ```

   The system response:

   ```
   Sample AMQSPUT0 start
   target queue is QTEST
   Sample AMQSPUT0 end
   ```

4. Get the test message using the sample program **amqsget**.

   ```
   amqsget QTEST QMGR
   ```

   The system response:

   ```
   Sample AMQSGET0 start
   message <A test message>
   Wait 15 seconds ...
   ```

```
no more messages
Sample AMQSGET0 end
```

5. Stop the queue manager.

```
endmqm -i QMGR
```

The system response:

```
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ended.
```

6. Delete the queue manager.

```
dltmqm QMGR
```

The system response:

```
IBM MQ queue manager 'QMGR' deleted.
```

7. Delete the directories you created.

   Tip: Add the /Q option to the commands to prevent the command prompting to delete each file or directory.

```
del /F /S C:\wmq\*.*
rmdir /S C:\wmq
```

Related tasks

Creating an Active Directory and DNS domain for IBM MQ

Installing IBM MQ on a server or workstation in a Windows domain

Creating a shared directory for queue manager data and log files

Create a multi-instance queue manager on domain controllers

An example shows how to set up a multi-instance queue manager on Windows on domain controllers. The setup demonstrates the concepts involved, rather than being production scale. The example is based on Windows Server 2008. The steps might differ on other versions of Windows Server.

The configuration uses the concept of a mini-domain, or "domainlet" ; see Windows 2000, Windows Server 2003, and Windows Server 2008 cluster nodes as domain controllers. To add multi-instance queue managers to an existing domain, see "Create a multi-instance queue manager on domain workstations or servers" on page 407.

The example configuration consists of three servers:

sun
: A Windows Server 2008 server used as the first domain controller. It defines the *wmq.example.com* domain that contains *sun* , *earth* , and *mars* . It contains one instance of the multi-instance queue manager called *QMGR* .

earth
: A Windows Server 2008 used as the second domain controller IBM MQ server. It contains the second instance of the multi-instance queue manager called *QMGR* .

mars
: A Windows Server 2008 used as the file server.

Replace the italicized names in the example, with names of your choosing.

Before you begin

1. On Windows, you do not need to verify the file system that you plan to store queue manager data and log files on. The checking procedure, Verifying shared file system behavior, is applicable to UNIX and Linux. On Windows, the checks are always successful.
2. Do the steps in "Creating an Active Directory and DNS domain for IBM MQ" on page 410 to create the first domain controller.
3. Do the steps in "Adding a second domain controller to the wmq.example.com domain" on page 425 to add a second domain controller, install IBM MQ for Windows on both domain controllers, and verify the installations.
4. Do the steps in "Installing IBM MQ on domain controllers in the wmq.example.com domain" on page 426 to install IBM MQ on the two domain controllers.

About this task

On a file server in the same domain create a share for the queue manager log and data directories. Next, create the first instance of a multi-instance queue manager that uses the file share on one of the domain controllers. Create the other instance on the other domain controller and finally verify the configuration. You can create the file share on a domain controller.

In the sample, *sun* is the first domain controller, *earth* the second, and *mars* is the file server.

Procedure

1. Create the directories that are to contain the queue manager data and log files.
   a) On *mars* , type the command:

   ```
   md c:\wmq\data , c:\wmq\logs
   ```

2. Share the directories that are to contain the queue manager data and log files.

   You must permit full control access to the domain local group mqm, and the user ID you use to create the queue manager. In the example, user IDs that are members of Domain Administrators have the authority to create queue managers.

   The file share must be on a server that is in the same domain as the domain controllers. In the example, the server *mars* is in the same domain as the domain controllers.

   a) In Windows Explorer, right-click ***c:\wmq*** > **Properties**.
   b) Click the **Security** tab and click **Advanced** > **Edit...**.
   c) Clear the check box for **Include inheritable permissions from this object's owner**. Click **Copy** in the Windows Security window.
   d) Select the lines for Users in the list of **Permission entries** and click **Remove**. Leave the lines for SYSTEM, Administrators, and CREATOR OWNER in the list of **Permission entries**.
   e) Click **Add...**, and type the name of the domain local group *mqm* . Click **Check Names**
   f) In response to a Windows Security window, Type the name and password of the Domain Administrator and click **OK** > **OK**.
   g) In the Permission Entry for wmq window, select **Full Control** in the list of **Permissions**.
   h) Click **OK** > **Apply** > **OK** > **OK** > **OK**
   i) Repeat steps e to h to add Domain Administrators.
   j) In Windows Explorer, right-click ***c:\wmq*** > **Share...**.
   k) Click **Advanced Sharing...** and select the **Share this folder** check box. Leave the share name as *wmq* .
   l) Click **Permissions** > **Add...**, and type the name of the domain local group *mqm* ; Domain Administrators. Click **Check Names**.

m) In response to a Windows Security window, Type the name and password of the `Domain Administrator` and click **OK** > **OK**.

3. Create the queue manager *QMGR* on the first domain controller, *sun* .

```
crtmqm -sax -u SYSTEM.DEAD.LETTER.QUEUE -md \\mars\wmq\data -ld \\mars\wmq\logs QMGR
```

The system response:

```
IBM MQ queue manager created.
Directory '\\mars\wmq\data\QMGR' created.
The queue manager is associated with installation 'Installation1'.
Creating or replacing default objects for queue manager 'QMGR'.
Default objects statistics : 74 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

4. Start the queue manager on *sun* , permitting a standby instance.

```
strmqm -x QMGR
```

The system response:

```
IBM MQ queue manager 'QMGR' starting.
The queue manager is associated with installation 'Installation1'.
5 log records accessed on queue manager 'QMGR' during the log
replay phase.
Log replay for queue manager 'QMGR' complete.
Transaction manager state recovered for queue manager 'QMGR'.
IBM MQ queue manager 'QMGR' started using V7.1.0.0.
```

5. Create a second instance of *QMGR* on *earth* .

a) Check the values of the `Prefix` and `InstallationName` parameters are correct for *earth* .

On *sun* , run the **dspmqinf** command:

```
dspmqinf QMGR
```

The system response:

```
QueueManager:
Name=QMGR
Directory=QMGR
Prefix=C:\ProgramData\IBM\MQ
DataPath=\\mars\wmq\data\QMGR
InstallationName=Installation1
```

b) Copy the machine-readable form of the **QueueManager** stanza to the clipboard.

On *sun* run the **dspmqinf** command again, with the `-o` command parameter.

```
dspmqinf -o command QMGR
```

The system response:

```
addmqinf -s QueueManager -v Name=QMGR
```

```
-v Directory=QMGR -v Prefix="C:\ProgramData\IBM\MQ"
-v DataPath=\\mars\wmq\data\QMGR
```

c) On *earth* run the **addmqinf** command from the clipboard to create an instance of the queue manager on *earth* .

Adjust the command, if necessary, to accommodate differences in the `Prefix` or `InstallationName` parameters.

```
addmqinf -s QueueManager -v Name= QMGR
-v Directory= QMGR -v Prefix="C:\Program Files\IBM\WebSphere MQ"
-v DataPath=\\mars\wmq\data\QMGR
```

```
IBM MQ configuration information added.
```

6. Start the standby instance of the queue manager on *earth* .

```
strmqm -x QMGR
```

The system response:

```
IBM MQ queue manager 'QMGR' starting.
The queue manager is associated with installation 'Installation1'.
A standby instance of queue manager 'QMGR' has been started. The active
instance is running elsewhere.
```

Results

Verify that the queue manager switches over from *sun* to *earth* :

1. On *sun* , run the command:

```
endmqm -i -r -s QMGR
```

The system response on *sun* :

```
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ended, permitting switchover to
a standby instance.
```

2. On *earth* repeatedly type the command:

```
dspmq
```

The system responses:

```
QMNAME(QMGR) STATUS(Running as standby)
QMNAME(QMGR) STATUS(Running as standby)
QMNAME(QMGR) STATUS(Running)
```

What to do next

To verify a multi-instance queue manager using sample programs; see "Verify the multi-instance queue manager on Windows" on page 428.

Related tasks

"Adding a second domain controller to the wmq.example.com domain" on page 425

"Installing IBM MQ on domain controllers in the wmq.example.com domain" on page 426

Related information

Windows 2000, Windows Server 2003, and Windows Server 2008 cluster nodes as domain controllers

Adding a second domain controller to the `wmq.example.com` domain

Add a second domain controller to the *`wmq.example.com`* domain to construct a Windows domain in which to run multi-instance queue managers on domain controllers and file servers.

The example configuration consists of three servers:

sun
: A Windows Server 2008 server used as the first domain controller. It defines the *`wmq.example.com`* domain that contains *`sun`*, *`earth`*, and *`mars`*. It contains one instance of the multi-instance queue manager called *`QMGR`*.

earth
: A Windows Server 2008 used as the second domain controller IBM MQ server. It contains the second instance of the multi-instance queue manager called *`QMGR`*.

mars
: A Windows Server 2008 used as the file server.

Replace the italicized names in the example, with names of your choosing.

Before you begin

1. Do the steps in "Creating an Active Directory and DNS domain for IBM MQ" on page 410 to create a domain controller, *`sun`*, for the domain *`wmq.example.com`*. Change the italicized names to suit your configuration.
2. Install Windows Server 2008 on a server in the default workgroup, `WORKGROUP`. For the example, the server is named *`earth`*.

About this task

In this task you configure a Windows Server 2008, called *`earth`*, as a second domain controller in the *`wmq.example.com`* domain.

This task is one of a set of related tasks that illustrate accessing queue manager data and log files. The tasks show how to create a queue manager authorized to read and write data and log files that are stored in a directory of your choosing. They accompany the task, "Windows domains and multi-instance queue managers" on page 406.

Procedure

1. Add the domain controller, *`sun.wmq.example.com`* to *`earth`* as a DNS server.
   a) On *`earth`*, log on as *`earth`*`\Administrator` and click **Start**.
   b) Right-click **Network** > **Properties** > **Manage network connections**.
   c) Right-click the network adapter, click **Properties**.

      The system responds with the Local Area Connection Properties window listing items the connection uses.
   d) Select the **Internet Protocol Version 4** or **Internet Protocol Version 6** from the list of items in the Local Area Connection Properties window. Click **Properties** > **Advanced...** and click the **DNS** tab.
   e) Under the DNS server addresses, click **Add...**.

f) Type the IP address of the domain controller, which is also the DNS server, and click **Add**.

g) Click **Append these DNS suffixes** > **Add...**.

h) Type *wmq.example.com* and click **Add**.

i) Type *wmq.example.com* in the **DNS suffix for this connection** field.

j) Select **Register this connection's address in DNS** and **Use this connection's suffix in DNS registration**. Click **OK** > **OK** > **Close**

k) Open a command window, and type the command **ipconfig /all** to review the TCP/IP settings.

2. Log on to the domain controller, *sun* , as the local or Workgroup administrator.

   If the server is already configured as a domain controller, you must log on as a domain administrator.

3. Run the Active Directory Domain Services wizard.

   a) Click **Start** > **Run...** Type dcpromo and click **OK**.

   If the Active Directory binary files are not already installed, Windows installs the files automatically.

4. Configure *earth* as the second domain controller in the *wmq.example.com* domain.

   a) In the first window of the wizard, leave the **Use advanced mode installation** check box clear. Click **Next** > **Next** and click **Create Add a domain controller to an existing domain** > **Next**.

   b) Type *wmq* into the **Type the name of any domain in this forest ...** field. The **Alternate credentials** radio button is clicked, click **Set...**. Type in the name and password of the domain administrator and click **OK** > **Next** > **Next** > **Next**.

   c) In the Additional Domain Controller Options window accept the **DNS server** and **Global catalog** options, which are selected; click **Next** > **Next**.

   d) On the Directory Services Restore Mode Administrator Password, type in a **Password** and **Confirm password** and click **Next** > **Next**.

   e) When prompted for **Network Credentials**, type in the password of the domain administrator. Select **Reboot on completion** in the final wizard window.

   f) After a while, a window might open with a **DCPromo** error concerning DNS delegation; click **OK**. The server reboots.

Results

When *earth* has rebooted, log on as Domain Administrator. Check that the wmq.example.com domain has been replicated to *earth* .

What to do next

Continue with installing IBM MQ ; see "Installing IBM MQ on domain controllers in the wmq.example.com domain" on page 426.

Related tasks

Installing IBM MQ on domain controllers in the wmq.example.com domain
"Creating an Active Directory and DNS domain for IBM MQ" on page 410

Installing IBM MQ on domain controllers in the wmq.example.com domain
Install and configure installations of IBM MQ on both domain controllers in the *wmq.example.com* domain.
Put your short description here; used for first paragraph and abstract.

The example configuration consists of three servers:

sun
: A Windows Server 2008 server used as the first domain controller. It defines the *wmq.example.com* domain that contains *sun* , *earth* , and *mars* . It contains one instance of the multi-instance queue manager called *QMGR* .

earth
A Windows Server 2008 used as the second domain controller IBM MQ server. It contains the second instance of the multi-instance queue manager called *QMGR* .

mars
A Windows Server 2008 used as the file server.

Replace the italicized names in the example, with names of your choosing.

Before you begin

1. Do the steps in "Creating an Active Directory and DNS domain for IBM MQ" on page 410 to create a domain controller, *sun* , for the domain *wmq.example.com* . Change the italicized names to suit your configuration.
2. Do the steps in "Adding a second domain controller to the wmq.example.com domain" on page 425 to create a second domain controller, *earth* , for the domain *wmq.example.com* . Change the italicized names to suit your configuration.
3. See Hardware and software requirements on Windows systems for other Windows versions you can run IBM MQ on.

About this task

Install and configure installations of IBM MQ on both domain controllers in the *wmq.example.com* domain.

Procedure

1. Install IBM MQ on *sun* and *earth* .

   For further information about running the IBM MQ for Windows installation wizard; see Installing IBM MQ server on Windows .

   a) On both *sun* and *earth* , log on as the domain administrator, *wmq*\Administrator.
   b) Run the **Setup** command on the IBM MQ for Windows installation media.

      The IBM MQ Launchpad application starts.
   c) Click **Software Requirements** to check that the prerequisite software is installed.
   d) Click **Network Configuration** > **No**.

      You can configure either a domain user ID or not for this installation. The user ID that is created is a domain local user ID.
   e) Click **IBM MQ Installation**, select an installation language and click Launch IBM MQ Installer.
   f) Confirm the license agreement and click **Next** > **Next** > **Install** to accept the default configuration. Wait for the installation to complete, and click **Finish**.

      If you want to change the name of the installation, install different components, configure a different directory for queue manager data and logs, or install into a different directory, click **Custom** rather than **Typical**.

      IBM MQ is installed, and the installer starts the "Prepare IBM MQ " wizard.

   The IBM MQ for Windows installation configures a domain local group mqm, and a domain group Domain mqm. It makes Domain mqm a member of mqm. Subsequent domain controllers in the same domain share the mqm and Domain mqm groups.
2. On both *earth* and *sun* , run the "Prepare IBM MQ " wizard.

   For further information about running the "Prepare IBM MQ " wizard, see Configuring IBM MQ with the Prepare IBM MQ wizard.

   a) The IBM MQ installer runs the "Prepare IBM MQ " automatically.

To start the wizard manually, find the shortcut to the "Prepare IBM MQ " in the **Start** > **All programs** > **IBM MQ** folder. Select the shortcut that corresponds to the installation of IBM MQ in a multi-installation configuration.

b) Click **Next** and leave **No** clicked in response to the question "Identify if there is a Windows 2000 or later domain controller in the network" [1] .

c) In the final page of the wizard, select or clear the check boxes as you require and click **Finish**.

The "Prepare IBM MQ " wizard creates a domain local user `MUSR_MQADMIN` on the first domain controller, and another domain local user `MUSR_MQADMIN1` on the second domain controller. The wizard creates the IBM MQ service on each controller, with `MUSR_MQADMIN` or `MUSR_MQADMIN1` as the user that logs on the service.

3. Define a user that has permission to create a queue manager.

   The user must have the right to log on locally, and be a member of the domain local `mqm` group. On domain controllers, domain users do not have the right to log on locally, but administrators do. By default, no user has both these attributes. In this task, add domain administrators to the domain local `mqm` group.

   a) Open **Server Manager** > **Roles** > **Active Directory Domain Services** > ***wmq.example.com*** > **Users**.

   b) Right-click **Domain Admins** > **Add to a group...** and type `mqm` ; click **Check names** > **OK** > **OK**

Results

1. Check that the "Prepare IBM MQ " created the domain user, `MUSR_MQADMIN`:
   a. Open **Server Manager** > **Roles** > **Active Directory Domain Services** > ***wmq.example.com*** > **Users**.
   b. Right-click **MUSR_MQADMIN** > **Properties...** > **Member Of**, and see that it is a member of `Domain users` and `mqm`.
2. Check that `MUSR_MQADMIN` has the right to run as a service:
   a. Click Click **Start** > **Run...**, type the command **`secpol.msc`** and click **OK**.
   b. Open **Security Settings** > **Local Policies** > **User Rights Assignments**. In the list of policies, right-click **Log on as a service** > **Properties** , and see `MUSR_MQADMIN` is listed as having the right to log on as a service. Click **OK**.

What to do next

1. Do the task, "Reading and writing data and log files authorized by the local mqm group" on page 436, to verify that the installation and configuration are working correctly.
2. Go back to the task, "Create a multi-instance queue manager on domain controllers" on page 421, to complete the task of configuring a multi-instance queue manager on domain controllers.

Related concepts

User rights required for an IBM MQ Windows Service

Related tasks

Adding a second domain controller to the wmq.example.com domain

Verify the multi-instance queue manager on Windows

Use the sample programs **amqsghac**, **amqsphac** and **amqsmhac** to verify a multi-instance queue manager configuration. This topic provides an example configuration to verify a multi-instance queue manager configuration on Windows Server 2003.

The high availability sample programs use automatic client reconnection. When the connected queue manager fails, the client attempts to reconnect to a queue manager in the same queue manager group. The description of the samples, High availability sample programs, demonstrates client reconnection

[1] You can configure the installation for the domain. As all users and groups on a domain controller have domain scope, it does not make any difference. It is simpler to install IBM MQ as if it is not in domain.

using a single instance queue manager for simplicity. You can use the same samples with multi-instance queue managers to verify a multi-instance queue manager configuration.

This example uses the multi-instance configuration described in "Create a multi-instance queue manager on domain controllers" on page 421. Use the configuration to verify that the multi-instance queue manager switches over to the standby instance. Stop the queue manager with the **endmqm** command and use the `-s`, switchover, option. The client programs reconnect to the new queue manager instance and continue to work with the new instance after a slight delay.

The client is installed in a 400 MB VMware image that is running Windows 7 Service Pack 1. For security reasons, it is connected on the same VMware host-only network as the domain servers that are running the multi-instance queue manager. It is sharing the `/MQHA` folder, which contains the client connection table, to simplify configuration.

Verifying failover using IBM MQ Explorer

Before using the sample applications to verify failover, run the IBM MQ Explorer on each server. Add both queue manager instances to each explorer using the **Add Remote Queue Manager > Connect directly to a multi-instance queue manager** wizard. Ensure that both instances are running, permitting standby. Close the window running the VMware image with the active instance, virtually powering off the server, or stop the active instance, allowing switchover to standby instance and reconnectable clients to reconnect.

Note: If you power off the server, make sure that it is not the one hosting the `MQHA` folder!

Note: The **Allow switchover to a standby instance** option might not be available on the **Stop Queue Manager** dialog. The option is missing because the queue manager is running as a single instance queue manager. You must have started it without the **Permit a standby instance** option. If your request to stop the queue manager is rejected, look at the **Details** window, possibly there is no standby instance running.

Verifying failover using the sample programs

Choose a server to run the active instance

You might have chosen one of the servers to host the `MQHA` directory or file system. If you plan to test failover by closing the VMware window running the active server, make sure that it is not the one hosting `MQHA` !

On the server running the active queue manager instance

1. Modify *ipaddr1* and *ipaddr2* and save the following commands in `N:\hasample.tst.`

```
DEFINE QLOCAL(SOURCE) REPLACE
DEFINE QLOCAL(TARGET) REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)  TRPTYPE(TCP) +
MCAUSER(' ') REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(' ipaddr1 (1414), ipaddr2 (1414)') QMNAME(QM1) REPLACE
START CHANNEL(CHANNEL1)
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) CONTROL(QMGR)
DISPLAY LISTENER(LISTENER.TCP) CONTROL
DISPLAY LSSTATUS(LISTENER.TCP) STATUS
```

   Note: By leaving the **MCAUSER** parameter blank, the client user ID is sent to the server. The client user ID must have the correct permissions on the servers. An alternative is to set the **MCAUSER** parameter in the `SVRCONN` channel to the user ID you have configured on the server.

2. Open a command prompt with the path `N:\` and run the command:

```
runmqsc -m QM1 < hasample.tst
```

3. Verify that the listener is running and has queue manager control, either by inspecting the output of the **runmqsc** command.

```
LISTENER(LISTENER.TCP)CONTROL(QMGR)
LISTENER(LISTENER.TCP)STATUS(RUNNING)
```

Or, using the IBM MQ Explorer that the TCPIP listener is running and has `Control = Queue Manager`.

On the client

1. Map the shared directory `C:\MQHA` on the server to `N:\` on the client.
2. Open a command prompt with the path `N:\`. Set the environment variable `MQCHLLIB` to point to the client channel definition table (CCDT) on the server:

   ```
   SET MQCHLLIB=N:\data\QM1\@ipcc
   ```

3. At the command prompt type the commands:

   ```
   start amqsghac TARGET QM1
   start amqsmhac -s SOURCE -t TARGET -m QM1
   start amqsphac SOURCE QM1
   ```

   Note: If you have problems, start the applications at a command prompt so that the reason code is printed out on the console, or look at the AMQERR01.LOG file in the `N:\data\QM1\errors` folder.

On the server running the active queue manager instance

1. Either:
   - Close the window running the VMware image with the active server instance.
   - Using the IBM MQ Explorer, stop the active queue manager instance, allowing switchover to the standby instance and instructing re-connectable clients to reconnect.
2. The three clients eventually detect the connection is broken, and then reconnect. In this configuration, if you close the server window, it is taking about seven minutes for all three connections to be reestablished. Some connections are reestablished well before others.

Results

```
N:\>amqsphac SOURCE QM1
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
17:05:25 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message <Message 6>
message <Message 7>
message <Message 8>
message <Message 9>
```

```
N:\>amqsmhac -s SOURCE -t TARGET -m QM1
Sample AMQSMHA0 start

17:05:25 : EVENT : Connection Reconnecting (Delay: 97ms)
17:05:48 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:53 : EVENT : Connection Reconnected
```

```
N:\>amqsghac TARGET QM1
Sample AMQSGHAC start
message <Message 1>
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
17:05:25 : EVENT : Connection Reconnecting (Delay: 156ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message <Message 6>
message <Message 7>
message <Message 8>
message <Message 9>
```

Secure shared queue manager data and log directories and files on Windows
This topic describes how you can secure a shared location for queue manager data and log files using a global alternative security group. You can share the location between different instances of a queue manager running on different servers.

Typically you do not set up a shared location for queue manager data and log files. When you install IBM MQ for Windows, the installation program creates a home directory of your choosing for any queue managers that are created on that server. It secures the directories with the local `mqm` group, and configures a user ID for the IBM MQ service to access the directories.

When you secure a shared folder with a security group, a user that is permitted to access the folder must have the credentials of the group. Suppose that a folder on a remote file server is secured with the local `mqm` group on a server called *mars* . Make the user that runs queue manager processes a member of the local `mqm` group on *mars* . The user has the credentials that match the credentials of the folder on the remote file server. Using those credentials, the queue manager is able to access its data and logs files in the folder. The user that runs queue manager processes on a different server is a member of a different local `mqm` group which does not have matching credentials. When the queue manager runs on a different server to *mars* , it cannot access the data and log files it created when it ran on *mars* . Even if you make the user a domain user, it has different credentials, because it must acquire the credentials from the local `mqm` group on *mars* , and it cannot do that from a different server.

Providing the queue manager with a global alternative security group solves the problem; see Figure 75 on page 432. Secure a remote folder with a global group. Pass the name of the global group to the queue manager when you create it on *mars* . Pass the global group name as the alternative security group using the `-a[r]` parameter on the **crtmqm** command. If you transfer the queue manager to run on a different server, the name of the security group is transferred with it. The name is transferred in the **AccessMode** stanza in the `qm.ini` file as a `SecurityGroup` ; for example:

```
AccessMode:
SecurityGroup=wmq\wmq
```

The **AccessMode** stanza in the `qm.ini` also includes the `RemoveMQMAccess` ; for example:

```
AccessMode:
RemoveMQMAccess=<true\false>
```

If this attribute is specified with value `true`, and an access group has also been given, the local mqm group is not granted access to the queue manager data files.

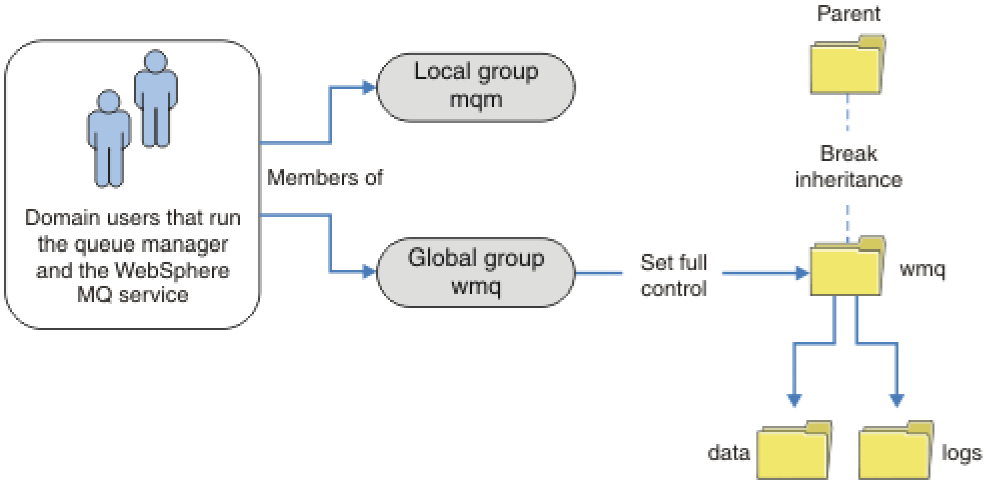

Figure 75. Securing queue manager data and logs using an alternative global security group (1)

For the user ID that queue manager processes are to run with to have the matching credentials of the global security group, the user ID must also have global scope. You cannot make a local group or principal a member of a global group. In Figure 75 on page 432, the users that run the queue manager processes are shown as domain users.

If you are deploying many IBM MQ servers, the grouping of users in Figure 75 on page 432 is not convenient. You would need to repeat the process of adding users to local groups for every IBM MQ server. Instead, create a `Domain mqm` global group on the domain controller, and make the users that run IBM MQ members of the `Domain mqm` group; see Figure 76 on page 432. When you install IBM MQ as a domain installation, the "Prepare IBM MQ " wizard automatically makes the `Domain mqm` group a member of the local `mqm` group. The same users are in both the global groups `Domain mqm` and `wmq`.

Tip: The same users can run IBM MQ on different servers, but on an individual server you must have different users to run IBM MQ as a service, and run interactively. You must also have different users for every installation on a server. Typically, therefore `Domain mqm` contains a number of users.

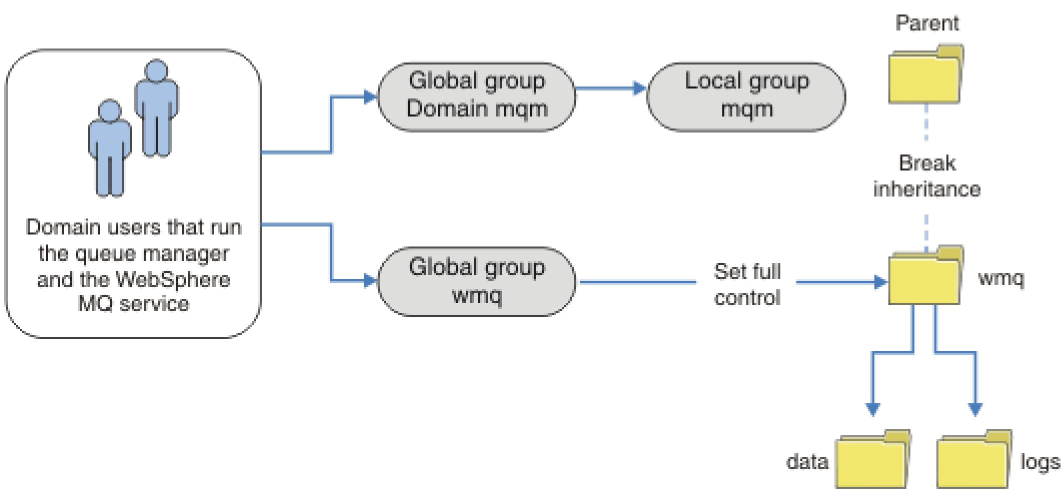

Figure 76. Securing queue manager data and logs using an alternative global security group (2)

The organization in Figure 76 on page 432 is unnecessarily complicated as it stands. The arrangement has two global groups with identical members. You might simplify the organization, and define only one global group; see Figure 77 on page 433.

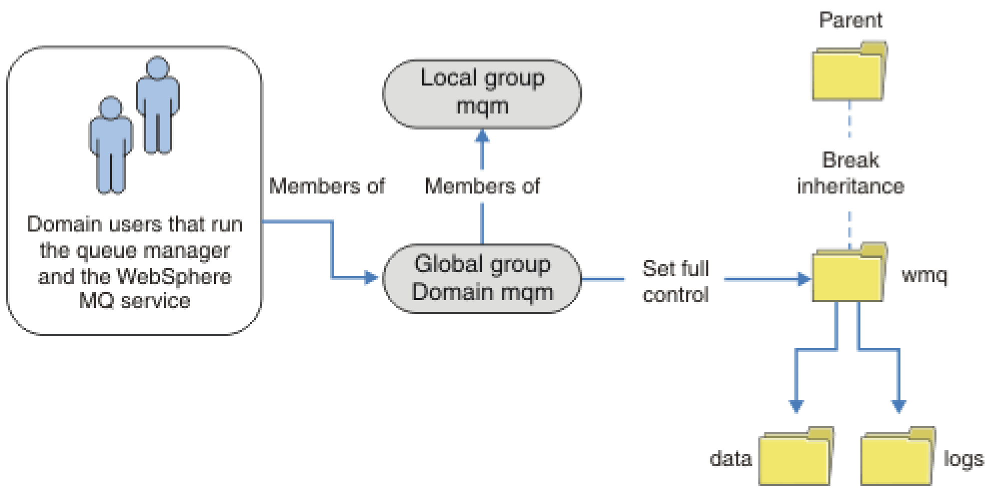

Figure 77. Securing queue manager data and logs using an alternative global security group (3)

Alternatively, you might need a finer degree of access control, with different queue managers restricted to being able to access different folders; see Figure 78 on page 434. In Figure 78 on page 434, two groups of domain users are defined, in separate global groups to secure different queue manager log and data files. Two different local `mqm` groups are shown, which must be on different IBM MQ servers. In this example, the queue managers are partitioned into two sets, with different users allocated to the two sets. The two sets might be test and production queue managers. The alternate security groups are called `wmq1` and `wmq2`. You must manually add the global groups `wmq1` and `wmq2` to the correct queue managers according to whether they are in the test or production department. The configuration cannot take advantage that the installation of IBM MQ propagates `Domain mqm` to the local `mqm` group as in Figure 77 on page 433, because there are two groups of users.

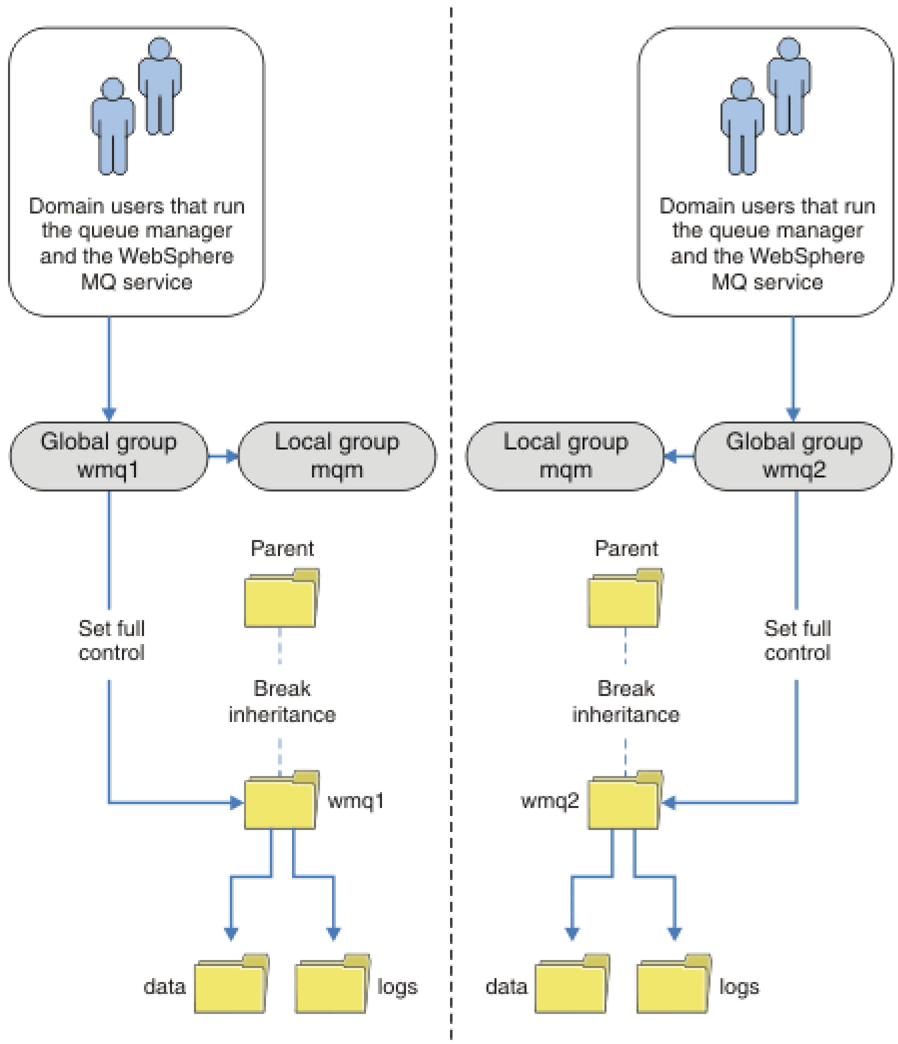

Figure 78. Securing queue manager data and logs using an alternative global security principal (4)

An alternative way to partition two departments would be to place them in two windows domains. In that case, you could return to using the simpler model shown in Figure 77 on page 433.

Secure unshared queue manager data and log directories and files on Windows
This topic describes how you can secure an alternative location for queue manager data and log files, both by using the local mqm group and an alternative security group.

Typically you do not set up an alternative location for queue manager data and log files. When you install IBM MQ for Windows, the installation program creates a home directory of your choosing for any queue managers that are created. It secures the directories with the local mqm group, and configures a user ID for the IBM MQ service to access the directories.

Two examples demonstrate how to configure access control for IBM MQ. The examples show how to create a queue manager with its data and logs in directories that are not on the data and log paths created by the installation. In the first example, "Reading and writing data and log files authorized by the local mqm group" on page 436, you permit access to the queue and log directories by authorizing by the local mqm group. The second example, "Reading and writing data and log files authorized by an alternative local security group" on page 439, differs in that access to the directories is authorized by an alternative security group. When the directories are accessed by a queue manager running on only one server, securing the data and log files with the alternative security group gives you the choice of securing different queue managers with different local groups or principals. When the directories are accessed by a queue manager running on different servers, such as with a multi-instance queue manager, securing the data and log files with the alternate security group is the only choice; see "Secure shared queue manager data and log directories and files on Windows" on page 431.

Configuring the security permissions of queue manager data and log files is not a common task on Windows. When you install IBM MQ for Windows, you either specify directories for queue manager data and logs, or accept the default directories. The installation program automatically secures these directories with the local mqm group, giving it full control permission. The installation process makes sure the user ID that runs queue managers is a member of the local mqm group. You can modify the other access permissions on the directories to meet your access requirements.

If you move the data and log files directory to new locations, you must configure the security of the new locations. You might change the location of the directories if you back up a queue manager and restore it onto a different computer, or if you change the queue manager to be a multi-instance queue manager. You have a choice of two ways of securing the queue manager data and log directories in their new location. You can secure the directories by restricting access to the local mqm group, or you can restrict access to any security group of your choosing.

It takes the least number of steps to secure the directories using the local mqm group. Set the permissions on the data and log directories to allow the local mqm group full control. A typical approach is to copy the existing set of permissions, removing inheritance from the parent. You can then remove or restrict the permissions of other principals.

If you run the queue manager under a different user ID to the service set up by the Prepare IBM MQ wizard, that user ID must be a member of the local mqm group. The task, "Reading and writing data and log files authorized by the local mqm group" on page 436, takes you through the steps.

You can also secure queue manager data and log files using an alternative security group. The process of securing the queue manager data and log files with the alternative security group has a number of steps that refer to Figure 79 on page 435. The local group, wmq, is an example of an alternative security group.

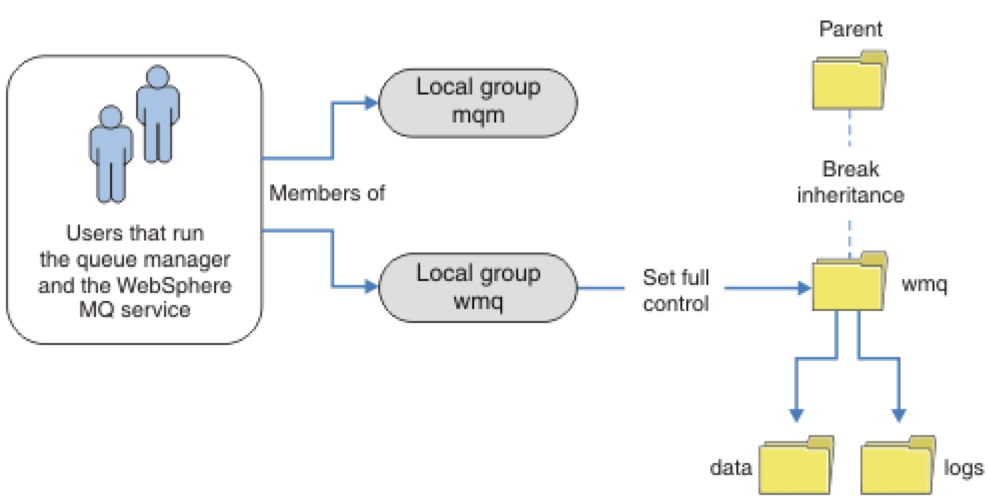

Figure 79. Securing queue manager data and logs using an alternative local security group, wmq

1. Either create separate directories for the queue manager data and logs, a common directory, or a common parent directory.
2. Copy the existing set of inherited permissions for the directories, or parent directory, and modify them according to your needs.
3. Secure the directories that are to contain the queue manager and logs by giving the alternative group, wmq, full control permission to the directories.
4. Give all user IDs that run queue manager processes the credentials of the alternative security group or principal:

a. If you define a user as the alternative security principal, the user must be same user as the queue manager is going to run under. The user must be a member of the local `mqm` group.
b. If you define a local group as the alternative security group, add the user that the queue manager is going to run under to the alternative group. The user must also be a member of the local `mqm` group.
c. If you define an global group as the alternative security group, then see "Secure shared queue manager data and log directories and files on Windows" on page 431.

5. Create the queue manager specifying the alternative security group or principal on the **crtmqm** command, with the `-a` parameter.

Reading and writing data and log files authorized by the local mqm group
The task illustrates how to create a queue manager with its data and logs files stored in any directory of your choosing. Access to the files is secured by the local `mqm` group. The directory is not shared.

Before you begin

1. Install IBM MQ for Windows as the primary installation.
2. Run the "Prepare IBM MQ " wizard. For this task, configure the installation either to run with a local user ID, or a domain user ID. Eventually, to complete all the tasks in "Windows domains and multi-instance queue managers" on page 406, the installation must be configured for a domain.
3. Log on with Administrator authority to perform the first part of the task.

About this task

This task is one of a set of related tasks that illustrate accessing queue manager data and log files. The tasks show how to create a queue manager authorized to read and write data and log files that are stored in a directory of your choosing. They accompany the task, "Windows domains and multi-instance queue managers" on page 406.

On Windows, you can create the default data and log paths for an IBM MQ for Windows in any directories of your choosing. The installation and configuration wizard automatically gives the local `mqm` group, and the user ID that is running the queue manager processes, access to the directories. If you create a queue manager specifying different directories for queue manager data and log files, you must configure full control permission to the directories.

In this example, you give the queue manager full control over its data and log files by giving the local `mqm` group permission to the directory *c:\wmq* .

The **crtmqm** command creates a queue manager that starts automatically when the workstation starts using the IBM MQ service.

The task is illustrative; it uses specific values that you can change. The values you can change are in italics. At the end of the task, follow the instructions to remove all the changes you made.

Procedure

1. Open a command prompt.
2. Type the command:

```
md c:\wmq\data, c:\wmq\logs
```

3. Set the permissions on the directories to allow the local `mqm` group read and write access.

```
cacls c:\wmq/T /E /G mqm:F
```

The system response:

```
processed dir: c:\wmq
```

```
processed dir: c:\wmq\data
processed dir: c:\wmq\logs
```

4. Optional: Switch to a user ID that is a member of the local `mqm` group.

   You can continue as Administrator, but for a realistic production configuration, continue with a user ID with more restricted rights. The user ID must at least be a member of the local `mqm` group.

   If the IBM MQ installation is configured as part of a domain, make the user ID a member of the `Domain mqm` group. The "Prepare IBM MQ " wizard makes the `Domain mqm` global group a member of the local `mqm` group, so you do not have to make the user ID directly a member of the local `mqm` group.
5. Create the queue manager.

   ```
   crtmqm -sax -u SYSTEM.DEAD.LETTER.QUEUE -md c:\wmq\data -ld c:\wmq\logs QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager created.
   Directory 'c:\wmq\data\QMGR' created.
   The queue manager is associated with installation '1'
   Creating or replacing default objects for queue manager 'QMGR'
   Default objects statistics : 74 created. 0 replaced.
   Completing setup.
   Setup completed.
   ```
6. Check that the directories created by the queue manager are in the *c:\wmq* directory.

   ```
   dir c:\wmq/D /B /S
   ```
7. Check that the files have read and write, or full control permission for the local `mqm` group.

   ```
   cacls c:\wmq\*.*
   ```

What to do next

Test the queue manager by putting and getting a message to a queue.

1. Start the queue manager.

   ```
   strmqm QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager 'QMGR' starting.
   The queue manager is associated with installation '1'.
   5 log records accessed on queue manager 'QMGR' during the log
   replay phase.
   Log replay for queue manager 'QMGR' complete.
   Transaction manager state recovered for queue manager 'QMGR'.
   IBM MQ queue manager 'QMGR' started using V7.1.0.0.
   ```
2. Create a test queue.

   ```
   echo define qlocal(QTEST) | runmqsc QMGR
   ```

   The system response:

```
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QMGR.


1 : define qlocal(QTEST)
AMQ8006: IBM MQ queue created.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

3. Put a test message using the sample program **amqsput**.

   ```
   echo 'A test message' | amqsput QTEST QMGR
   ```

   The system response:

   ```
   Sample AMQSPUT0 start
   target queue is QTEST
   Sample AMQSPUT0 end
   ```

4. Get the test message using the sample program **amqsget**.

   ```
   amqsget QTEST QMGR
   ```

   The system response:

   ```
   Sample AMQSGET0 start
   message <A test message>
   Wait 15 seconds ...
   no more messages
   Sample AMQSGET0 end
   ```

5. Stop the queue manager.

   ```
   endmqm -i QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager 'QMGR' ending.
   IBM MQ queue manager 'QMGR' ended.
   ```

6. Delete the queue manager.

   ```
   dltmqm QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager 'QMGR' deleted.
   ```

7. Delete the directories you created.

   Tip: Add the /Q option to the commands to prevent the command prompting to delete each file or directory.

   ```
   del /F /S C:\wmq\*.*
   rmdir /S C:\wmq
   ```

Related concepts

“Windows domains and multi-instance queue managers” on page 406

A multi-instance queue manager on Windows requires its data and logs to be shared. The share must be accessible to all instances of the queue manager running on different servers or workstations. Configure the queue managers and share as part of a Windows domain. The queue manager can run on a domain workstation or server, or on the domain controller.

Related tasks

Reading and writing data and log files authorized by an alternative local security group

This task shows how to use the -a flag on the **crtmqm** command. The flag provides the queue manager with an alternative local security group to give it access to its log and data files.

“Reading and writing shared data and log files authorized by an alternative global security group” on page 418

“Create a multi-instance queue manager on domain workstations or servers” on page 407

Reading and writing data and log files authorized by an alternative local security group

This task shows how to use the -a flag on the **crtmqm** command. The flag provides the queue manager with an alternative local security group to give it access to its log and data files.

Before you begin

1. Install IBM MQ for Windows as the primary installation.
2. Run the "Prepare IBM MQ " wizard. For this task, configure the installation either to run with a local user ID, or a domain user ID. Eventually, to complete all the tasks in “Windows domains and multi-instance queue managers” on page 406, the installation must be configured for a domain.
3. Log on with Administrator authority to perform the first part of the task.

About this task

This task is one of a set of related tasks that illustrate accessing queue manager data and log files. The tasks show how to create a queue manager authorized to read and write data and log files that are stored in a directory of your choosing. They accompany the task, “Windows domains and multi-instance queue managers” on page 406.

On Windows, you can create the default data and log paths for an IBM MQ for Windows in any directories of your choosing. The installation and configuration wizard automatically gives the local mqm group, and the user ID that is running the queue manager processes, access to the directories. If you create a queue manager specifying different directories for queue manager data and log files, you must configure full control permission to the directories.

In this example, you provide the queue manager with an alternative security local group that has full control authorization to the directories. The alternative security group gives the queue manager permission to manage files in the directory. The primary purpose of the alternate security group is to authorize an alternate security global group. Use an alternate security global group to set up a multi-instance queue manager. In this example, you configure a local group to familiarize yourself with the use of an alternate security group without installing IBM MQ in a domain. It is unusual to configure a local group as an alternative security group.

The **crtmqm** command creates a queue manager that starts automatically when the workstation starts using the IBM MQ service.

The task is illustrative; it uses specific values that you can change. The values you can change are in italics. At the end of the task, follow the instructions to remove all the changes you made.

Procedure

1. Set up an alternative security group.

The alternative security group is typically a domain group. In the example, you create a queue manager that uses a local alternate security group. With a local alternate security group, you can do the task with an IBM MQ installation that is not part of a domain.

a) Run the **lusrmgr.msc** command to open the Local Users and Groups window.

b) Right-click **Groups** > **New Group...**

c) In the **Group name** field, type *altmqm* and click **Create** > **Close**.

d) Identify the user ID that runs the IBM MQ service.

  i) Click **Start** > **Run...**, type `services.msc` and click **OK**.

  ii) Click the IBM MQ service in the list of services, and click the Log On tab.

  iii) Remember the user ID and close the Services Explorer.

e) Add the user ID that runs the IBM MQ service to the *altmqm* group. Also add the user ID that you log on with to create a queue manager, and run it interactively.

  Windows checks the authority of the queue manager to access the data and logs directories by checking the authority of the user ID that is running queue manager processes. The user ID must be a member, directly or indirectly through a global group, of the *altmqm* group that authorized the directories.

  If you installed IBM MQ as part of a domain, and are going to do the tasks in "Create a multi-instance queue manager on domain workstations or servers" on page 407, the domain user IDs created in "Creating an Active Directory and DNS domain for IBM MQ" on page 410 are *wmquser1* and *wmquser2* .

  If you did not install the queue manager as part of a domain, the default local user ID that runs the IBM MQ service is `MUSR_MQADMIN`. If you intend to do the tasks without Administrator authority, create a user that is a member of the local `mqm` group.

  Follow these steps to add *wmquser1* and *wmquser2* to *altmqm* . If your configuration is different, substitute your names for the user IDs and group.

  i) In the list of groups right-click **altmqm** > **Properties** > **Add...**.

  ii) In the Select Users, Computers, or Groups window type *wmquser1* ; *wmquser2* and click **Check Names**.

  iii) Type the name and password of a domain administrator in the Windows Security window, then click **OK** > **OK** > **Apply** > **OK**.

2. Open a command prompt.
3. Restart the IBM MQ service.

   You must restart the service so that the user ID it runs under acquires the additional security credentials you configured for it.

   Type the commands:

   ```
   endmqsvc
   strmqsvc
   ```

   The system responses:

   ```
   5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
   The MQ service for installation 'Installation1' ended successfully.
   ```

   And:

   ```
   5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
   The MQ service for installation 'Installation1' started successfully.
   ```
4. Type the command:

```
md c:\wmq\data, c:\wmq\logs
```

5. Set the permissions on the directories to allow the local user *user* read and write access.

   ```
   cacls c:\wmq/T /E /G altmqm:F
   ```

   The system response:

   ```
   processed dir: c:\wmq
   processed dir: c:\wmq\data
   processed dir: c:\wmq\logs
   ```

6. Optional: Switch to a user ID that is a member of the local mqm group.

   You can continue as Administrator, but for a realistic production configuration, continue with a user ID with more restricted rights. The user ID must at least be a member of the local mqm group.

   If the IBM MQ installation is configured as part of a domain, make the user ID a member of the Domain mqm group. The "Prepare IBM MQ " wizard makes the Domain mqm global group a member of the local mqm group, so you do not have to make the user ID directly a member of the local mqm group.

7. Create the queue manager.

   ```
   crtmqm -a altmqm -sax -u SYSTEM.DEAD.LETTER.QUEUE -md c:\wmq\data -ld c:\wmq\logs QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager created.
   Directory 'c:\wmq1\data\QMGR' created.
   The queue manager is associated with installation '1'
   Creating or replacing default objects for queue manager 'QMGR'
   Default objects statistics : 74 created. 0 replaced.
   Completing setup.
   Setup completed.
   ```

8. Check that the directories created by the queue manager are in the *c:\wmq* directory.

   ```
   dir c:\wmq/D /B /S
   ```

9. Check that the files have read and write, or full control permission for the local mqm group.

   ```
   cacls c:\wmq\*.*
   ```

What to do next

Test the queue manager by putting and getting a message to a queue.

1. Start the queue manager.

   ```
   strmqm QMGR
   ```

   The system response:

   ```
   IBM MQ queue manager 'QMGR' starting.
   The queue manager is associated with installation '1'.
   5 log records accessed on queue manager 'QMGR' during the log
   ```

```
replay phase.
Log replay for queue manager 'QMGR' complete.
Transaction manager state recovered for queue manager 'QMGR'.
IBM MQ queue manager 'QMGR' started using V7.1.0.0.
```

2. Create a test queue.

```
echo define qlocal(QTEST) | runmqsc QMGR
```

The system response:

```
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QMGR.


1 : define qlocal(QTEST)
AMQ8006: IBM MQ queue created.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

3. Put a test message using the sample program **amqsput**.

```
echo 'A test message' | amqsput QTEST QMGR
```

The system response:

```
Sample AMQSPUT0 start
target queue is QTEST
Sample AMQSPUT0 end
```

4. Get the test message using the sample program **amqsget**.

```
amqsget QTEST QMGR
```

The system response:

```
Sample AMQSGET0 start
message <A test message>
Wait 15 seconds ...
no more messages
Sample AMQSGET0 end
```

5. Stop the queue manager.

```
endmqm -i QMGR
```

The system response:

```
IBM MQ queue manager 'QMGR' ending.
IBM MQ queue manager 'QMGR' ended.
```

6. Delete the queue manager.

```
dltmqm QMGR
```

The system response:

```
IBM MQ queue manager 'QMGR' deleted.
```

7. Delete the directories you created.

   Tip: Add the `/Q` option to the commands to prevent the command prompting to delete each file or directory.

   ```
   del /F /S C:\wmq\*.*
   rmdir /S C:\wmq
   ```

Related tasks

Reading and writing data and log files authorized by the local mqm group
The task illustrates how to create a queue manager with its data and logs files stored in any directory of your choosing. Access to the files is secured by the local `mqm` group. The directory is not shared.

Create a multi-instance queue manager on Linux
An example shows how to set up a multi-instance queue manager on Linux. The setup is small to illustrate the concepts involved. The example is based on Linux Red Hat Enterprise 5. The steps differ on other UNIX platforms.

About this task

The example is set up on a 2 GHz notebook computer with 3 GB RAM running Windows 7 Service Pack 1. Two VMware virtual machines, Server1 and Server2, run Linux Red Hat Enterprise 5 in 640 MB images. Server1 hosts the network file system (NFS), the queue manager logs and an HA instance. It is not usual practice for the NFS server also to host one of the queue manager instances; this is to simplify the example. Server2 mounts Server1's queue manager logs with a standby instance. A WebSphere MQ MQI client is installed on an additional 400 MB VMware image that runs Windows 7 Service Pack 1 and runs the sample high availability applications. All the virtual machines are configured as part of a VMware host-only network for security reasons.

Note: You should put only queue manager data on an NFS server. On the NFS, use the following three options with the mount command to make the system secure:

noexec
By using this option, you stop binary files from being run on the NFS, which prevents a remote user from running unwanted code on the system.

nosuid
By using this option, you prevent the use of the set-user-identifier and set-group-identifier bits, which prevents a remote user from gaining higher privileges.

nodev
By using this option, you stop character and block special devices from being used or defined, which prevents a remote user from getting out of a chroot jail.

Procedure

1. Log in as root.
2. Follow the instructions in Installing IBM MQ to install IBM MQ, create the mqm user and group, and define `/var/mqm`.
3. Complete the task Verifying shared file system behavior to check that the file system supports multi-instance queue managers.
4. For Server1, complete the following step:
   a. Create log and data directories in a common folder, `/MQHA`, that is to be shared. For example:
      i) **mkdir** */MQHA*
      ii) **mkdir** */MQHA/logs*
      iii) **mkdir** */MQHA/qmgrs*

5. For Server2, complete the following step:
   a. Create the folder, `/MQHA`, to mount the shared file system. Keep the path the same as on Server1. For example:
      i) **mkdir** */MQHA*
6. Ensure that the MQHA directories are owned by user and group mqm, and the access permissions are set to `rwx` for user and group. For example **ls -al** displays `drwxrwxr-x mqm mqm 4096 Nov 27 14:38 MQDATA.`
   a. **chown -R** `mqm:mqm` */MQHA*
   b. **chmod -R** `ug+rwx` */MQHA*
7. Create the queue manager by entering the following command: **crtmqm -ld** */MQHA/logs* **-md** */MQHA/qmgrs QM1*
8. Add [2] `/MQHA *(rw,sync,no_wdelay,fsid=0)` to `/etc/exports`
9. For Server1, complete the following steps:
   a. Start the NFS daemon: */etc/init.d/* **nfs** `start`
   b. Copy the queue manager configuration details from Server1:

```
dspmqinf -o command QM1
```

   and copy the result to the clipboard:

```
addmqinf -s QueueManager
-v Name=QM1
-v Directory=QM1
-v Prefix=/var/mqm
-v DataPath=/MQHA/qmgrs/QM1
```

10. For Server2, complete the following steps:
    a. Mount the exported file system `/MQHA` by entering the following command: **mount -t** `nfs4` **-o** `hard,intr` *Server1:/ /MQHA*
    b. Paste the queue manager configuration command into Server2:

```
addmqinf -s QueueManager
-v Name=QM1
-v Directory=QM1
-v Prefix=/var/mqm
-v DataPath=/MQHA/qmgrs/QM1
```

11. Start the queue manager instances, in either order, with the **-x** parameter: **strmqm -x** `QM1`.

    The command used to start the queue manager instances must be issued from the same IBM MQ installation as the **addmqinf** command. To start and stop the queue manager from a different installation, you must first set the installation associated with the queue manager using the **setmqm** command. For more information, see setmqm.

Verifying the multi-instance queue manager on Linux
Use the sample programs **amqsghac**, **amqsphac** and **amqsmhac** to verify a multi-instance queue manager configuration. This topic provides an example configuration to verify a multi-instance queue manager configuration on Linux Red Hat Enterprise 5.

The high availability sample programs use automatic client reconnection. When the connected queue manager fails, the client attempts to reconnect to a queue manager in the same queue manager group. The description of the samples, High availability sample programs, demonstrates client reconnection

[2] The `'*'` allows all machines that can reach this one mount /MQHA for read/write. Restrict access on a production machine.

using a single instance queue manager for simplicity. You can use the same samples with multi-instance queue managers to verify a multi-instance queue manager configuration.

The example uses the multi-instance configuration described in "Create a multi-instance queue manager on Linux" on page 443. Use the configuration to verify that the multi-instance queue manager switches over to the standby instance. Stop the queue manager with the **endmqm** command and use the `-s`, switchover, option. The client programs reconnect to the new queue manager instance and continue to work with the new instance after a slight delay.

In the example, the client is running on a Windows 7 Service Pack 1 system. The system is hosting two VMware Linux servers that are running the multi-instance queue manager.

Verifying failover using IBM MQ Explorer

Before using the sample applications to verify failover, run the IBM MQ Explorer on each server. Add both queue manager instances to each explorer using the **Add Remote Queue Manager > Connect directly to a multi-instance queue manager** wizard. Ensure that both instances are running, permitting standby. Close the window running the VMware image with the active instance, virtually powering off the server, or stop the active instance, allowing switchover to standby instance.

Note: If you power off the server, make sure that it is not the one hosting `/MQHA` !

Note: The **Allow switchover to a standby instance** option might not be available on the **Stop Queue Manager** dialog. The option is missing because the queue manager is running as a single instance queue manager. You must have started it without the **Permit a standby instance** option. If your request to stop the queue manager is rejected, look at the **Details** window, it is possibly because there is no standby instance running.

Verifying failover using the sample programs

Choose a server to be to run the active instance

You might have chosen one of the servers to host the `MQHA` directory or file system. If you plan to test failover by closing the VMware window running the active server, make sure that it is not the one hosting `MQHA` !

On the server running the active queue manager instance

Note: Running the `SVRCONN` channel with the `MCAUSER` set to `mqm`, is a convenience to reduce the number of configuration steps in the example. If another user ID is chosen, and your system is set up differently to the one used in the example, you might experience access permission problems. Do not use `mqm` as a `MCAUSER` on an exposed system; it is likely to compromise security greatly.

1. Modify *ipaddr1* and *ipaddr2* and save the following commands in `/MQHA/hasamples.tst.`

```
DEFINE QLOCAL(SOURCE) REPLACE
DEFINE QLOCAL(TARGET) REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)  TRPTYPE(TCP) +
MCAUSER('mqm') REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(' ipaddr1 (1414), ipaddr2
(1414)') QMNAME(QM1) REPLACE
START  CHANNEL(CHANNEL1)
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) CONTROL(QMGR)
DISPLAY LISTENER(LISTENER.TCP) CONTROL
START  LISTENER(LISTENER.TCP)
DISPLAY LSSTATUS(LISTENER.TCP) STATUS
```

2. Open a terminal window with the path `/MQHA` and run the command:

```
runmqsc -m QM1 < hasamples.tst
```

3. Verify that the listener is running and has queue manager control, either by inspecting the output of the **runmqsc** command.

```
LISTENER(LISTENER.TCP)CONTROL(QMGR)
LISTENER(LISTENER.TCP)STATUS(RUNNING)
```

   Or, using the IBM MQ Explorer that the TCPIP listener is running and has `Control = Queue Manager`.

On the client

1. Copy the client connection table AMQCLCHL.TAB from `/MQHA/qmgrs/QM1.000/@ipcc` on the server to `C:\` on the client.
2. Open a command prompt with the path `C:\` and set the environment variable `MQCHLLIB` to point to the client channel definition table (CCDT)

```
SET MQCHLLIB=C:\
```

3. At the command prompt type the commands:

```
start amqsghac TARGET QM1
start amqsmhac -s SOURCE -t TARGET -m QM1
start amqsphac SOURCE QM1
```

On the server running the active queue manager instance

1. Either:
   - Close the window running the VMware image with the active server instance.
   - Using the IBM MQ Explorer, stop the active queue manager instance, allowing switchover to the standby instance and instructing reconnectable clients to reconnect.
2. The three clients eventually detect the connection is broken, and then reconnect. In this configuration, if you close the server window, it is taking about seven minutes for all three connections to be reestablished. Some connections are reestablished well before others.

Results

```
N:\>amqsphac SOURCE QM1
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
17:05:25 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message <Message 6>
message <Message 7>
message <Message 8>
message <Message 9>
```

```
N:\>amqsmhac -s SOURCE -t TARGET -m QM1
Sample AMQSMHA0 start

17:05:25 : EVENT : Connection Reconnecting (Delay: 97ms)
17:05:48 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:53 : EVENT : Connection Reconnected
```

```
N:\>amqsghac TARGET QM1
Sample AMQSGHAC start
message <Message 1>
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
17:05:25 : EVENT : Connection Reconnecting (Delay: 156ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message <Message 6>
message <Message 7>
message <Message 8>
message <Message 9>
```

Deleting a multi-instance queue manager

To delete a multi-instance queue manager completely, you need to use the **dltmqm** command to delete the queue manager, and then remove instances from other servers using either the **rmvmqinf** or **dltmqm** commands.

Run the **dltmqm** command to delete a queue manager that has instances defined on other servers, on any server where that queue manager is defined. You do not need to run the **dltmqm** command on the same server that you created it on. Then run the **rmvmqinf** or **dltmqm** command on all the other servers which have a definition of the queue manager.

You can only delete a queue manager when it is stopped. At the time you delete it no instances are running, and the queue manager, strictly speaking, is neither a single or a multi-instance queue manager; it is simply a queue manager that has its queue manager data and logs on a remote share. When you delete a queue manager, its queue manager data and logs are deleted, and the queue manager stanza is removed from the `mqs.ini` file on the server on which you issued the **dltmqm** command. You need to have access to the network share containing the queue manager data and logs when you delete the queue manager.

On other servers where you have previously created instances of the queue manager there are also entries in the `mqs.ini` files on those servers. You need to visit each server in turn, and remove the queue manager stanza by running the command **rmvmqinf** *Queue manager stanza name*.

On UNIX and Linux systems, if you have placed a common `mqs.ini` file in network storage and referenced it from all the servers by setting the `AMQ_MQS_INI_LOCATION` environment variable on each server, then you need to delete the queue manager from only one of its servers as there is only one `mqs.ini` file to update.

Example

First server
: **dltmqm** `QM1`

Other servers where instances are defined
: **rmvmqinf** `QM1` , or

: **dltmqm** `QM1`

Starting and stopping a multi-instance queue manager

Starting and stopping a queue manager configured either as a single instance or a multi-instance queue manager.

When you have defined a multi-instance queue manager on a pair of servers, you can run the queue manager on either server, either as a single instance queue manager, or as a multi-instance queue manager.

To run a multi-instance queue manager, start the queue manager on one of the servers using the **strmqm** `-x` *QM1* command; the `-x` option permits the instance to failover. It becomes the *active instance*. Start the standby instance on the other server using the same **strmqm** `-x` *QM1* command; the `-x` option permits the instance to start as a standby.

The queue manager is now running with one active instance that is processing all requests, and a standby instance that is ready to take over if the active instance fails. The active instance is granted exclusive access to the queue manager data and logs. The standby waits to be granted exclusive access to the queue manager data and logs. When the standby is granted exclusive access, it becomes the active instance.

You can also manually switch control to the standby instance by issuing the **endmqm** `-s` command on the active instance. The **endmqm** `-s` command shuts down the active instance without shutting down the standby. The exclusive access lock on the queue manager data and logs is released, and the standby takes over.

You can also start and stop a queue manager configured with multiple instances on different servers as a single instance queue manager. If you start the queue manager without using the `-x` option on the **strmqm** command, the instances of the queue manager configured on other machines are prevented from starting as standby instances. If you attempt to start another instance you receive the response that the queue manager instance is not permitted to run as a standby.

If you stop the active instance of a multi-instance queue manager using the **endmqm** command without the `-s` option, then the active and standby instances both stop. If you stop the standby instance using the **endmqm** command with the `-x` option, then it stops being a standby, and the active instance continues running. You cannot issue **endmqm** without the `-x` option on the standby.

Only two queue manager instances can run at the same time; one is the active instance, and the other is a standby instance. If you start two instances at the same time, IBM MQ has no control over which instance becomes the active instance; it is determined by the network file system. The first instance to acquire exclusive access to the queue manager data becomes the active instance.

Note: Before you restart a failed queue manager, you must disconnect your applications from that instance of the queue manager. If you do not, the queue manager might not restart correctly.

Shared file system

A multi-instance queue manager uses a networked file system to manage queue manager instances.

A multi-instance queue manager automates failover using a combination of file system locks and shared queue manager data and logs. Only one instance of a queue manager can have exclusive access to the shared queue manager data and logs. When it gets access it becomes the active instance. The other instance that does not succeed in getting exclusive access waits as a standby instance until the queue manager data and logs become available.

The networked file system is responsible for releasing the locks it holds for the active queue manager instance. If the active instance fails in some way, the networked file system releases the locks it is holding for the active instance. As soon as the exclusive lock is released, a standby queue manager waiting for the lock attempts to acquire it. If it succeeds, it becomes the active instance and has exclusive access to the queue manager data and logs on the shared file system. It then continues to start.

The related topic, Planning file system support describes how to set up and check that your file system supports multi-instance queue managers.

A multi-instance queue manager does not protect you against a failure in the file system. There are a number of ways to protect your data.

- Invest in reliable storage, such as redundant disk arrays (RAID), and include them in a networked file system that has network resilience.
- Back up IBM MQ linear logs to alternative media, and if your primary log media fails, recover using the logs on the alternative media. You can use a backup queue manager to administer this process.

Multiple queue manager instances

A multi-instance queue manager is resilient because it uses a standby queue manager instance to restore queue manager availability after failure.

Replicating queue manager instances is a very effective way to improve the availability of queue manager processes. Using a simple availability model, purely for illustration: if the reliability of one instance of a queue manager is 99% (over one year, cumulative downtime is 3.65 days) then adding another instance of the queue manager increases the availability to 99.99% (over one year, cumulative downtime of about an hour).

This is too simple a model to give you practical numeric estimates of availability. To model availability realistically, you need to collect statistics for the mean time between failures (MTBF) and the mean time to repair (MTTR), and the probability distribution of time between failures and of repair times.

The term, multi-instance queue manager, refers to the combination of active and standby instances of the queue manager that share the queue manager data and logs. Multi-instance queue managers protect you against the failure of queue manager processes by having one instance of the queue manager active on one server, and another instance of the queue manager on standby on another server, ready to take over automatically should the active instance fail.

Failover or switchover

A standby queue manager instance takes over from the active instance either on request (switchover), or when the active instance fails (failover).

- *Switchover* takes place when a standby instance starts in response to the **endmqm** `-s` command being issued to the active queue manager instance. You can specify the **endmqm** parameters `-c`, `-i` or `-p` to control how abruptly the queue manager is stopped.

  Note: Switchover only takes place if a standby queue manager instance is already started. The **endmqm** `-s` command releases the active queue manager lock and permits switchover: it does not start a standby queue manager instance.

- *Failover* occurs when the lock on queue manager data held by the active instance is released because the instance appeared to stop unexpectedly (that is, without an **endmqm** command being issued).

When the standby instance takes over as the active instance, it writes a message to the queue manager error log.

Reconnectable clients are automatically reconnected when a queue manager fails or switches over. You do not need to include the `-r` flag on the **endmqm** command to request client reconnection. Automatic client reconnect is not supported by IBM MQ classes for Java.

If you find that you cannot restart a failed instance, even though failover has occurred and the standby instance has become active, check to see whether applications connected locally to the failed instance have disconnected from the failed instance.

Locally connected applications must end or disconnect from a failed queue manager instance in order for the failed instance to be restarted. Any locally connected applications using shared bindings (which is the default setting) which hold on to a connection to a failed instance act to prevent the instance from being restarted.

If it is not possible to end the locally connected applications, or ensure that they disconnect when the local queue manager instance fails, consider using isolated bindings. Locally connected applications using isolated bindings do not prevent the local queue manager instance from being restarted, even if they do not disconnect.

Channel and client reconnection

Channel and client reconnection is an essential part of restoring message processing after a standby queue manager instance has become active.

Multi-instance queue manager instances are installed on servers with different network addresses. You need to configure IBM MQ channels and clients with connection information for all queue manager instances. When a standby takes over, clients and channels are automatically reconnected to the newly active queue manager instance at the new network address. Automatic client reconnect is not supported by IBM MQ classes for Java.

The design is different from the way high availability environments such as HA-CMP work. HA-CMP provides a virtual IP address for the cluster and transfer the address to the active server. IBM MQ reconnection does not change or reroute IP addresses. It works by reconnecting using the network addresses you have defined in channel definitions and client connections. As an administrator, you need to define the network addresses in channel definitions and client connections to all instances of any multi-instance queue manager. The best way to configure network addresses to a multi-instance queue manager depends on the connection:

Queue manager channels
: The `CONNAME` attribute of channels is a comma-separated list of connection names; for example, `CONNAME('127.0.0.1(1234), 192.0.2.0(4321)')`. The connections are tried in the order specified in the connection list until a connection is successfully established. If no connection is successful, the channel attempts to reconnect.

Cluster channels
: Typically, no additional configuration is required to make multi-instance queue managers work in a cluster.

  If a queue manager connects to a repository queue manager, the repository discovers the network address of the queue manager. It refers to the `CONNAME` of the `CLUSRCVR` channel at the queue manager. On TCPIP, the queue manager automatically sets the `CONNAME` if you omit it, or configure it to blanks. When a standby instance takes over, its IP address replaces the IP address of the previous active instance as the `CONNAME`.

  If it is necessary, you can manually configure `CONNAME` with the list of network addresses of the queue manager instances.

Client connections
: Client connections can use connection lists, or queue manager groups to select alternative connections. Clients need to be compiled to run with IBM WebSphere MQ 7.0.1 client libraries or better. They must be connected to at least a Version 7.0.1 queue manager.

When failover occurs, reconnection takes some time. The standby queue manager has to complete its startup. The clients that were connected to the failed queue manager have to detect the connection failure, and start a new client connection. If a new client connection selects the standby queue manager that has become newly active, then the client is reconnected to the same queue manager.

If the client is in the middle of an MQI call during the reconnection, it must tolerate an extended wait before the call completes.

If the failure takes place during a batch transfer on a message channel, the batch is rolled back and restarted.

Switching over is faster than failing over, and takes only as long as stopping one instance of the queue manager and starting another. For a queue manager with only few log records to replay, at best switchover might take of the order of a few seconds. To estimate how long failover takes, you need to add the time that it takes for the failure to be detected. At best the detection takes of the order of 10 seconds, and might be several minutes, depending on the network and the file system.

Application recovery

Application recovery is the automated continuation of application processing after failover. Application recovery following failover requires careful design. Some applications need to be aware failover has taken place.

The objective of application recovery is for the application to continue processing with only a short delay. Before continuing with new processing, the application must back out and resubmit the unit of work that it was processing during the failure.

A problem for application recovery is losing the context that is shared between the IBM MQ MQI client and the queue manager, and stored in the queue manager. The IBM MQ MQI client restores most of the context, but there are some parts of the context that cannot be reliably restored. The following sections describe some properties of application recovery and how they affect the recovery of applications connected to a multi-instance queue manager.

Transactional messaging

From the perspective of delivering messages, failover does not change the persistent properties of IBM MQ messaging. If messages are persistent, and correctly managed within units of work, then messages are not lost during a failover.

From the perspective of transaction processing, transactions are either backed out or committed after failover.

Uncommitted transactions are rolled back. After failover, a re-connectable application receives a `MQRC_BACKED_OUT` reason code to indicate that the transaction has failed. It then needs to restart the transaction again.

Committed transactions are transactions that have reached the second phase of a two-phase commit, or single phase (message only) transactions that have begun `MQCMIT`.

If the queue manager is the transaction coordinator and `MQCMIT` has begun the second phase of its two-phase commit before the failure, the transaction successfully completes. The completion is under the control of the queue manager and continues when the queue manager is running again. In a reconnectable application, the `MQCMIT` call completes normally.

In a single phase commit, which involves only messages, a transaction that has started commit processing completes normally under the control of the queue manager once it is running again. In a reconnectable application, the `MQCMIT` completes normally.

Reconnectable clients can use single phase transactions under the control of the queue manager as the transaction coordinator. The extended transactional client does not support reconnection. If reconnection is requested when the transactional client connects, the connection succeeds, but without the ability to be reconnected. The connection behaves as if it is not reconnectable.

Application restart or resume

Failover interrupts an application. After a failure an application can restart from the beginning, or it can resume processing following the interruption. The latter is called *automatic client reconnection*. Automatic client reconnect is not supported by IBM MQ classes for Java.

With an IBM MQ MQI client application, you can set a connection option to reconnect the client automatically. The options are `MQCNO_RECONNECT` or `MQCNO_RECONNECT_Q_MGR`. If no option is set, the client does not try to reconnect automatically and the queue manager failure returns `MQRC_CONNECTION_BROKEN` to the client. You might design the client to try and start a new connection by issuing a new `MQCONN` or `MQCONNX` call.

Server programs have to be restarted; they cannot be automatically reconnected by the queue manager at the point they were processing when the queue manager or server failed. IBM MQ server programs are typically not restarted on the standby queue manager instance when a multi-instance queue manager instance fails.

You can automate an IBM MQ server program to restart on the standby server in two ways:

1. Package your server application as a queue manager service. It is restarted when the standby queue manager restarts.
2. Write your own failover logic, triggered for example, by the failover log message written by a standby queue manager instance when it starts. The application instance then needs to call `MQCONN` or `MQCONNX` after it starts, to create a connection to the queue manager.

Detecting failover

Some applications do need to be aware of failover, others do not. Consider these two examples.

1. A messaging application that gets or receives messages over a messaging channel does not normally require the queue manager at the other end of the channel to be running: it is unlikely to be affected if the queue manager at the other end of the channel restarts on a standby instance.

2. An IBM MQ MQI client application processes persistent message input from one queue and puts persistent message responses onto another queue as part of a single unit of work: if it handles an MQRC_BACKED_OUT reason code from MQPUT, MQGET, or MQCMIT within sync point by restarting the unit of work, then no messages are lost. Additionally the application does not need to do any special processing to deal with a connection failure.

Suppose however, in the second example, that the application is browsing the queue to select the message to process by using the MQGET option, MQGMO_MSG_UNDER_CURSOR. Reconnection resets the browse cursor, and the MQGET call does not return the correct message. In this example, the application has to be aware failover has occurred. Additionally, before issuing another MQGET for the message under the cursor, the application must restore the browse cursor.

Losing the browse cursor is one example of how the application context changes following reconnection. Other cases are documented in "Recovery of an automatically reconnected client" on page 452.

You have three alternative design patterns for IBM MQ MQI client applications following failover. Only one of them does not need to detect the failover.

No reconnection

In this pattern, the application stops all processing on the current connection when the connection is broken. For the application to continue processing, it must establish a new connection with the queue manager. The application is entirely responsible for transferring any state information it requires to continue processing on the new connection. Existing client applications that reconnect with a queue manager after losing their connection are written in this way.

The client receives a reason code, such as MQRC_CONNECTION_BROKEN, or MQRC_Q_MGR_NOT_AVAILABLE from the next MQI call after the connection is lost. The application must discard all its IBM MQ state information, such as queue handles, and issue a new MQCONN or MQCONNX call to establish a new connection, and then reopen the IBM MQ objects it needs to process.

The default MQI behavior is for the queue manager connection handle to become unusable after a connection with the queue manager is lost. The default is equivalent to setting the MQCNO_RECONNECT_DISABLED option on MQCONNX to prevent application reconnection after failover.

Failover tolerant

Write the application so it is unaffected by failover. Sometimes careful error handling is sufficient to deal with failover.

Reconnection aware

Register an MQCBT_EVENT_HANDLER event handler with the queue manager. The event handler is posted with MQRC_RECONNECTING when the client starts to try to reconnect to the server, and MQRC_RECONNECTED after a successful reconnection. You can then run a routine to reestablish a predictable state so that the client application is able to continue processing.

Recovery of an automatically reconnected client

Failover is an unexpected event, and for an automatically reconnected client to work as designed the consequences of reconnection must be predictable.

A major element of turning an unexpected failure into a predictable and reliable recovery is the use of transactions.

In the previous section, an example, "2" on page 452, was given of an IBM MQ MQI client using a local transaction to coordinate MQGET and MQPUT. The client issues an MQCMIT or MQBACK call in response to a MQRC_BACKED_OUT error and then resubmits the backed out transaction. The queue manager failure causes the transaction to be backed out, and the behavior of the client application ensures no transactions, and no messages, are lost.

Not all program state is managed as part of a transaction, and therefore the consequences of reconnection become harder to understand. You need to know how reconnection changes the state of an IBM MQ MQI client in order to design your client application to survive queue manager failover.

You might decide to design your application without any special failover code, handling reconnection errors with the same logic as other errors. Alternatively, you might choose to recognize that reconnection requires special error processing, and register an event handler with IBM MQ to run a routine to handle failover. The routine might handle the reconnection processing itself, or set a flag to indicate to the main program thread that when it resumes processing it needs to perform recovery processing.

The IBM MQ MQI client environment is aware of failover itself, and restores as much context as it can, following reconnection, by storing some state information in the client, and issuing additional MQI calls on behalf of the client application to restore its IBM MQ state. For example, handles to objects that were open at the point of failure are restored, and temporary dynamic queues are opened with the same name. But there are changes that are unavoidable and you need your design to deal with these changes. The changes can be categorized into five kinds:

1. New, or previously undiagnosed errors, are returned from MQI calls until a consistent new context state is restored by the application program.

   An example of receiving a new error is the return code `MQRC_CONTEXT_NOT_AVAILABLE` when trying to pass context after saving context before the reconnection. The context cannot be restored after reconnection because the security context is not passed to an unauthorized client program. To do so would let a malicious application program obtain the security context.

   Typically, applications handle common and predictable errors in a carefully designed way, and relegate uncommon errors to a generic error handler. The error handler might disconnect from IBM MQ and reconnect again, or even stop the program altogether. To improve continuity, you might need to deal with some errors in a different way.
2. Non-persistent messages might be lost.
3. Transactions are rolled back.
4. `MQGET` or `MQPUT` calls used outside a sync point might be interrupted with the possible loss of a message.
5. Timing induced errors, due to a prolonged wait in an MQI call.

Some details about lost context are listed in the following section.

- Non-persistent messages are discarded, unless put to a queue with the `NPMCLASS(HIGH)` option, and the queue manager failure did not interrupt the option of storing non-persistent messages on shutdown.
- A non-durable subscription is lost when a connection is broken. On reconnection, it is re-established. Consider using a durable subscription.
- The get-wait interval is recomputed; if its limit is exceeded it returns `MQRC_NO_MSG_AVAILABLE`. Similarly, subscription expiry is recomputed to give the same overall expiry time.
- The position of the browse cursor in a queue is lost; it is typically reestablished before the first message.
  - `MQGET` calls that specify `MQGMO_BROWSE_MSG_UNDER_CURSOR` or `MQGMO_MSG_UNDER_CURSOR`, fail with reason code `MQRC_NO_MSG_AVAILABLE`.
  - Messages locked for browsing are unlocked.
  - Browse marked messages with handle scope are unmarked and can be browsed again.
  - Cooperatively browse marked messages are unmarked in most cases.
- Security context is lost. Attempts to use saved message context, such as putting a message with `MQPMO_PASS_ALL_CONTEXT` fail with `MQRC_CONTEXT_NOT_AVAILABLE`.
- Message tokens are lost. `MQGET` using a message token returns the reason code `MQRC_NO_MSG_AVAILABLE`.

  Note: *MsgId* and *CorrelId*, as they are part of the message, are preserved with the message during failover, and so `MQGET` using `MsgId` or `CorrelId` work as expected.
- Messages put on a queue under sync point in an uncommitted transaction are no longer available.

- Processing messages in a logical order, or in a message group, results in a return code of `MQRC_RECONNECT_INCOMPATIBLE` after reconnection.
- An MQI call might return `MQRC_RECONNECT_FAILED` rather than the more general `MQRC_CONNECTION_BROKEN` that clients typically receive today.
- Reconnection during an `MQPUT` call outside sync point returns `MQRC_CALL_INTERRUPTED` if the IBM MQ MQI client does not know if the message was delivered to the queue manager successfully. Reconnection during `MQCMIT` behaves similarly.
- `MQRC_CALL_INTERRUPTED` is returned - after a successful reconnect - if the IBM MQ MQI client has received no response from the queue manager to indicate the success or failure of
  - the delivery of a persistent message using an MQPUT call outside sync point.
  - the delivery of a persistent message or a message with default persistence using an MQPUT1 call outside sync point.
  - the commit of a transaction using an MQCMIT call. The response is only ever returned after a successful reconnect.
- Channels are restarted as new instances (they might also be different channels), and so no channel exit state is retained.
- Temporary dynamic queues are restored as part of the process of recovering reconnectable clients that had temporary dynamic queues open. No messages on a temporary dynamic queue are restored, but applications that had the queue open, or had remembered the name of the queue, are able to continue processing.

  There is the possibility that if the queue is being used by an application other than the one that created it, that it might not be restored quickly enough to be present when it is next referenced. For example, if a client creates a temporary dynamic queue as a reply-to queue, and a reply message is to be placed on the queue by a channel, the queue might not be recovered in time. In this case, the channel would typically place the reply-to message on the dead letter queue.

  If a reconnectable client application opens a temporary dynamic queue by name (because another application has already created it), then when reconnection occurs, the IBM MQ MQI client is unable to recreate the temporary dynamic queue because it does not have the model to create it from. In the MQI, only one application can open the temporary dynamic queue by model. Other applications that wish to use the temporary dynamic queue must use MQPUT1, or server bindings, or be able to try the reconnection again if it fails.

  Only non-persistent messages might be put to a temporary dynamic queue, and these messages are lost during failover; this loss is true for messages being put to a temporary dynamic queue using MQPUT1 during reconnection. If failover occurs during the MQPUT1, the message might not be put, although the MQPUT1 succeeds. One workaround to this problem is to use permanent dynamic queues. Any server bindings application can open the temporary dynamic queue by name because it is not reconnectable.

Data recovery and high availability

High availability solutions using multi-instance queue managers must include a mechanism to recover data after a storage failure.

A multi-instance queue manager increases the availability of queue manager processes, but not the availability of other components, such as the file system, that the queue manager uses to store messages, and other information.

One way to make data highly available is to use networked resilient data storage. You can either build your own solution using a networked file system and resilient data storage, or you can buy an integrated solution. If you want to combine resilience with disaster recovery, then asynchronous disk replication, which permits disk replication over tens, or hundreds of kilometers, is available.

You can configure the way different IBM MQ directories are mapped to storage media, to make the best use of the media. For *multi-instance* queue managers there is an important distinction between two types of IBM MQ directories and files.

Directories that must be shared between the instances of a queue manager.

The information that must be shared between different instances of a queue manager is in two directories: the `qmgrs` and `logs` directories. The directories must be on a shared networked file system. You are advised to use a storage media that provides continuous high availability and excellent performance because the data is constantly changing as messages are created and deleted.

Directories and files that do not *have* to be shared between instances of a queue manager.

Some other directories do not have to be shared between different instances of a queue manager, and are quickly restored by means other than using a mirrored file system.

- IBM MQ executable files and the tools directory. Replace by reinstalling or by backing up and restoring from a backed up file archive.
- Configuration information that is modified for the installation as a whole. The configuration information is either managed by IBM MQ, such as the `mqs.ini` file on Windows, UNIX and Linux systems, or part of your own configuration management such as **`MQSC`** configuration scripts. Back up and restore using a file archive.
- Installation-wide output such as traces, error logs and FFDC files. The files are stored in the `errors` and `trace` subdirectories in the default data directory. The default data directory on UNIX and Linux systems is `/var/mqm`. On Windows the default data directory is the IBM MQ installation directory.

You can also use a backup queue manager to take regular media backups of a multi-instance queue manager using linear logging. A backup queue manager does not provide recovery that is as fast as from a mirrored file system, and it does not recover changes since the last backup. The backup queue manager mechanism is more appropriate for use in off-site disaster recovery scenarios than recovering a queue manager after a localized storage failure.

Combining IBM MQ Availability solutions

Applications are using other IBM MQ capabilities to improve availability. Multi-instance queue managers complement other high availability capabilities.

IBM MQ Clusters increase queue availability

You can increase queue availability by creating multiple definitions of a cluster queue; up to one of every queue on each manager in the cluster.

Suppose a member of the cluster fails and then a new message is sent to a cluster queue. Unless the message *has* to go to the queue manager that has failed, the message is sent to another running queue manager in the cluster that has a definition of the queue.

Although clusters greatly increase availability, there are two related failure scenarios that result in messages getting delayed. Building a cluster with multi-instance queue managers reduces the chance of a message being delayed.

Marooned messages

If a queue manager in the cluster fails, no more messages that can be routed to other queue managers in the cluster are routed to the failed queue manager. Messages that have already been sent are marooned until the failed queue manager is restarted.

Affinities

Affinity is the term used to describe information shared between two otherwise separate computations. For example, an affinity exists between an application sending a request message to a server and the same application expecting to process the reply. Another example would be a sequence of messages, the processing of each message depending on the previous messages.

If you send messages to clustered queues you need to consider affinities. Do you need to send successive messages to the same queue manager, or can each message go to any member of the cluster?

If you do need to send messages to the same queue manager in the cluster and it fails, the messages wait in the transmission queue of the sender until the failed cluster queue manager is running again.

If the cluster is configured with multi-instance queue managers the delay waiting for the failed queue manager to restart is limited to the order of a minute or so while the standby takes over. When the standby is running, marooned messages resume processing, channels to the newly activated queue manager instance are started, and the messages that were waiting in transmission queues start flowing.

A possible way to configure a cluster to overcome messages being delayed by a failed queue manager, is to deploy two different queue managers to each server in the cluster, and arrange for one to be the active and one to be the standby instance of the different queue managers. This is an active-standby configuration, and it increases the availability of the cluster.

As well as having the benefits of reduced administration and increased scalability, clusters continue to provide additional elements of availability to complement multi-instance queue managers. Clusters protect against other types of failure that affect both the active and standby instances of a queue manager.

Uninterrupted service
: A cluster provides an uninterrupted service. New messages received by the cluster are sent to active queue managers to be processed. Do not rely on a multi-instance queue manager to provide an uninterrupted service because it takes time for the standby queue manager to detect the failure and complete its startup, for its channels to be reconnected, and for failed batches of messages to be resubmitted.

Localized outage
: There are practical limitations to how far apart the active, standby, and file system servers can be, as they need to interact at millisecond speeds to deliver acceptable performance.

Clustered queue managers require interaction speeds of the order of many seconds, and can be geographically dispersed anywhere in the world.

Operational error
: By using two different mechanisms to increase availability you reduce the chances that an operational error, such as a human error, compromises your availability efforts.

Queue sharing groups increase message processing availability

Queue sharing groups, provided only on z/OS, allow a group of queue managers to share servicing a queue. If one queue manager fails, the other queue managers continue to process all the messages on the queue. Multi-instance queue managers are not supported on z/OS and complement queue sharing groups only as part of a wider messaging architecture.

IBM MQ Clients increase application availability

IBM MQ MQI client programs can connect to different queue managers in a queue manager group based on queue manager availability, connection weightings, and affinities. By running an application on a different machine from the one on which the queue manager is running, you can to improve the overall availability of a solution as long as there is a way to reconnect the application if the queue manager instance it is connected to fails.

Queue manager groups are used to increase client availability by uncoupling a client from a queue manager that is stopped, and load balancing client connections across a group of queue managers, rather like an IP sprayer. The client application must have no affinities with the failed queue manager, such as a dependency on a particular queue, or it cannot resume processing.

Automatic client reconnection and multi-instance queue managers increase client availability by resolving some affinity problems. Automatic client reconnect is not supported by IBM MQ classes for Java.

You can set the `MQCNO` option `MQCNO_RECONNECT_Q_MGR`, to force a client to reconnect to the same queue manager:

1. If the previously connected single instance queue manager is not running the connection is tried again until the queue manager is running again.
2. If the queue manager is configured as a multi-instance queue manager, then the client reconnects to whichever instance is active.

By automatically reconnecting to the same queue manager, much of the state information the queue manager was holding on behalf of the client, such as the queues it had open and the topic it was subscribed to, are restored. If the client had opened a dynamic reply-to queue to receive a reply to a request, the connection to the reply-to queue is restored too.

Making sure that messages are not lost (logging)

IBM MQ records all significant changes to the persistent data controlled by the queue manager in a recovery log.

This includes creating and deleting objects, persistent message updates, transaction states, changes to object attributes, and channel activities. The log contains the information you need to recover all updates to message queues by:

- Keeping records of queue manager changes
- Keeping records of queue updates for use by the restart process
- Enabling you to restore data after a hardware or software failure

However, IBM MQ also relies on the disk system hosting its files, including log files. If the disk system is itself unreliable, information, including log information, can still be lost.

What logs look like

Logs consist of primary and secondary files, and a control file. You define the number and size of log files and where they are stored in the file system.

An IBM MQ log consists of two components:

1. One or more files of log data.
2. A log control file

A file of log data is also known as a log extent.

There are a number of log extents that contain the data being recorded. You can define the number and size (as explained in "Log defaults for IBM MQ" on page 99), or take the system default of three primary and two secondary extents.

Each of the three primary and two secondary extents defaults to 16 MB.

When you create a queue manager, the number of log extents pre-allocated is the number of *primary* log extents allocated. If you do not specify a number, the default value is used.

IBM MQ uses two types of logging:

- Circular
- Linear

  The number of log extents used with linear logging can be very large, depending on the frequency of your media image recording.

See "Types of logging" on page 458 for more information.

In IBM MQ for Windows, if you have not changed the log path, log extents are created under the directory:

```
C:\ProgramData\IBM\MQ\log\<QMgrName>
```

In IBM MQ for UNIX and Linux systems, if you have not changed the log path, log extents are created under the directory:

```
/var/mqm/log/<QMgrName>
```

IBM MQ starts with these primary log extents, but if the primary log space is not sufficient, it allocates *secondary* log extents. It does this dynamically and removes them when the demand for log space reduces. By default, up to two secondary log extents can be allocated. You can change this default allocation, as described in "Changing IBM MQ and queue manager configuration information" on page 80.

The log control file

The log control file contains information needed to describe the state of log extents, such as their size and location and the name of the next available extent.

Important: The log control file is for internal queue manager use only.

The queue manager keeps control data associated with the state of the recovery log in the log control file and you must not modify the contents of the log control file.

The log control file is in the log path and is called `amqhlctl.lfh`. When backing up or restoring your queue manager, ensure that the log control file is backed up and restored, along with your log extents.

Types of logging

In IBM MQ there are two ways of maintaining records of queue manager activities: circular logging and linear logging.

Circular logging

Use circular logging if all you want is restart recovery, using the log to roll back transactions that were in progress when the system stopped.

Circular logging keeps all restart data in a ring of log files. Logging fills the first file in the ring, then moves on to the next, and so on, until all the files are full. It then goes back to the first file in the ring and starts again. This continues as long as the product is in use, and has the advantage that you never run out of log files.

IBM MQ keeps the log entries required to restart the queue manager without loss of data until they are no longer required to ensure queue manager data recovery. The mechanism for releasing log files for reuse is described in "Restart recovery" on page 459.

Linear logging

Use linear logging if you want both restart recovery and media recovery (re-creating lost or damaged data by replaying the contents of the log). Linear logging keeps the log data in a continuous sequence of log files.

Log files can optionally be:

- Reused, but only when they are no longer needed for either restart recovery or media recovery.
- Manually archived for longer term storage and analysis.

The frequency of media images determines when linear log files can be reused, and is a major factor in how much disk space must be available for linear log files.

You can configure the queue manager to automatically take periodic media images, based either upon time or log usage, or you can schedule media images manually.

Your administrator decides what policy to implement, and the implications on disk space usage. Log files needed for restart recovery must always be available, whereas log files needed only for media recovery can be archived to longer term storage, for example, tape.

If your administrator enables automatic log management and automatic media images, linear logging behaves in a similar way to a very large circular log, but with the improved redundancy against media failure enabled by media recovery.

Active log

There are a number of files that are said to be *active* in both linear and circular logging. The active log is the maximum amount of log space, whether you are using circular or linear logging, that might be referenced by restart recovery.

The number of active log files is usually less than the number of primary log files as defined in the configuration files. (See "Calculating the size of the log" on page 463 for information about defining the number.)

Note, that the active log space does not include the space required for media recovery, and that the number of log files used with linear logging can be very large, depending on your message flow and the frequency of media images.

Inactive log

When a log file is no longer needed for restart recovery it becomes `inactive`. Log files that are not required for either restart recovery, or media recovery, can be considered as superfluous log files.

When using automatic log management, the queue manager controls the processing of these superfluous log files. If you have selected manual log management, it becomes the responsibility of your administrator to manage (for example, delete and archive) superfluous log files if they are no longer of interest to your operation.

Refer to "Managing logs" on page 466 for further information about the disposition of log files.

Secondary log files

Although secondary log files are defined for linear logging, they are not used in normal operation. If a situation arises when, probably due to long-lived transactions, it is not possible to free a file from the active pool because it might still be required for a restart, secondary files are formatted and added to the active log file pool.

If the number of secondary files available is used up, requests for most further operations requiring log activity will be refused with an MQRC_RESOURCE_PROBLEM return code being returned to the application, and any long running transactions will be considered for asynchronous rollback.

Attention: Both types of logging can cope with unexpected loss of power, assuming that there is no hardware failure.

Restart recovery

Both circular logging and linear logging queue managers support restart recovery. Regardless of how abruptly the previous instance of the queue manager terminates (for example a power outage) upon restart the queue manager restores its persistent state to the correct transactional state at the point of termination.

Restart recovery depends upon disk integrity being maintained. Similarly, the operating system should ensure disk integrity regardless of how abruptly an operating system termination might occur.

In the highly unusual event that disk integrity is not maintained then linear logging (and media recovery) provides some further redundancy and recoverability options. With increasingly common technology, such as RAID, it is increasingly rare to suffer disk integrity issues and many enterprises configure circular logging and use only restart recovery.

IBM MQ is designed as a classic Write Ahead Logging resource manager. Persistent updates to message queues happen in two stages:

1. Log records representing the update are written reliably to the recovery log
2. The queue file or buffers are updated in a manner that is the most efficient for your system, but not necessarily consistently.

The log files can thus become more up to date than the underlying queue buffer and file state.

If this situation was allowed to continue unabated, then a very large volume of log replay would be required to make the queue state consistent following a crash recovery.

IBM MQ uses `checkpoints` in order to limit the volume of log replay required following a crash recovery. The key event that controls whether a log file is termed active or not is a `checkpoint`.

An IBM MQ checkpoint is a point:

- Of consistency between the recovery log and object files.
- That identifies a place in the log, from which forward replay of subsequent log records is guaranteed to restore the queue to the correct logical state at the time the queue manager might have ended.

During a checkpoint, IBM MQ flushes older updates to the queues files, as required, in order to limit the volume of log records that need to be replayed to bring the queues back to a consistent state following a crash recovery.

The most recent complete checkpoint marks a point in the log from which replay must be performed during crash recovery. The frequency of checkpoint is thus a trade-off between the overhead of recording checkpoints, and the improvement in potential recovery time implied by those checkpoints.

The position in the log of the start of the most recent complete checkpoint is one of the key factors in determining whether a log file is active or inactive. The other key factor is the position in the log of the first log record relating to the first persistent update made by a current active transaction.

If a new checkpoint is recorded in the second, or later, log file and no current transaction refers to a log record in the first log file, the first log file become inactive. In the case of circular logging the first log file is now ready to be reused. In the case of linear logging the first log file will typically still be required for media recovery.

If you configure either circular logging or automatic log management the queue manager will manage the inactive log files. If you configure linear logging with manual log management it becomes an administrative task to manage the inactive files according to the requirements of your operation.

IBM MQ generates checkpoints automatically. They are taken when:

- The queue manager starts
- At shutdown
- When logging space is running low
- After 50,000 operations have been logged since the previous checkpoint was taken
- For z/OS, the LOGLOAD setting controls how many operations are in a checkpoint.

When IBM MQ restarts, it finds the latest checkpoint record in the log. This information is held in the checkpoint file that is updated at the end of every checkpoint. All the operations that have taken place since the checkpoint are replayed forward. This is known as the replay phase.

The replay phase brings the queues back to the logical state they were in before the system failure or shutdown. During the replay phase a list is created of the transactions that were in-flight when the system failure or shutdown occurred.

Messages AMQ7229 and AMQ7230 are issued to indicate the progression of the replay phase.

In order to know which operations to back out or commit, IBM MQ accesses each active log record associated with an in-flight transaction. This is known as the recovery phase.

Messages AMQ7231, AMQ7232 and AMQ7234 are issued to indicate the progression of the recovery phase.

Once all the necessary log records have been accessed during the recovery phase, each active transaction is in turn resolved and each operation associated with the transaction will be either backed out or committed. This is known as the resolution phase.

distributed Message AMQ7233 is issued to indicate the progression of the resolution phase.

z/OS On z/OS, restart processing is made up of various phases.

1. The recovery log range is established, based on the media recovery required for the page sets and the oldest log record that is required for backing out units of work and obtaining locks for in-doubt units of work.
2. Once the log range has been determined, forward log reading is carried out to bring the page sets up to the latest state, and also to lock any messages that are related to in-doubt or in-flight units of work.
3. When forward log reading has been completed the logs are read backwards to backout any units of work that were in-flight or in-backout at the time of failure.

z/OS An example of the messages you might see:

```
CSQR001I +MQOX RESTART INITIATED
CSQR003I +MQOX RESTART - PRIOR CHECKPOINT RBA=00000001E48C0A5E
CSQR004I +MQOX RESTART - UR COUNTS -  806
IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN BACKOUT=0
CSQR030I +MQOX Forward recovery log range  815
from RBA=00000001E45FF7AD to RBA=00000001E48C1882
CSQR005I +MQOX RESTART - FORWARD RECOVERY COMPLETE -  816
IN COMMIT=0, INDOUBT=0
CSQR032I +MQOX Backward recovery log range  817
from RBA=00000001E48C1882 to RBA=00000001E48C1882
CSQR006I +MQOX RESTART - BACKWARD RECOVERY COMPLETE -  818
INFLIGHT=0, IN BACKOUT=0
CSQR002I +MQOX RESTART COMPLETED
```

Note: If there is a large amount of log to be read, messages CSQR031I (forward recovery) and CSQR033I (backwards recovery) are issued periodically to show the progression.

In Figure 80 on page 462, all records before the latest checkpoint, Checkpoint 2, are no longer needed by IBM MQ. The queues can be recovered from the checkpoint information and any later log entries. For circular logging, any freed files before the checkpoint can be reused. For a linear log, the freed log files no longer need to be accessed for normal operation and become inactive. In the example, the queue head pointer is moved to point at the latest checkpoint, Checkpoint 2, which then becomes the new queue head, Head 2. Log File 1 can now be reused.

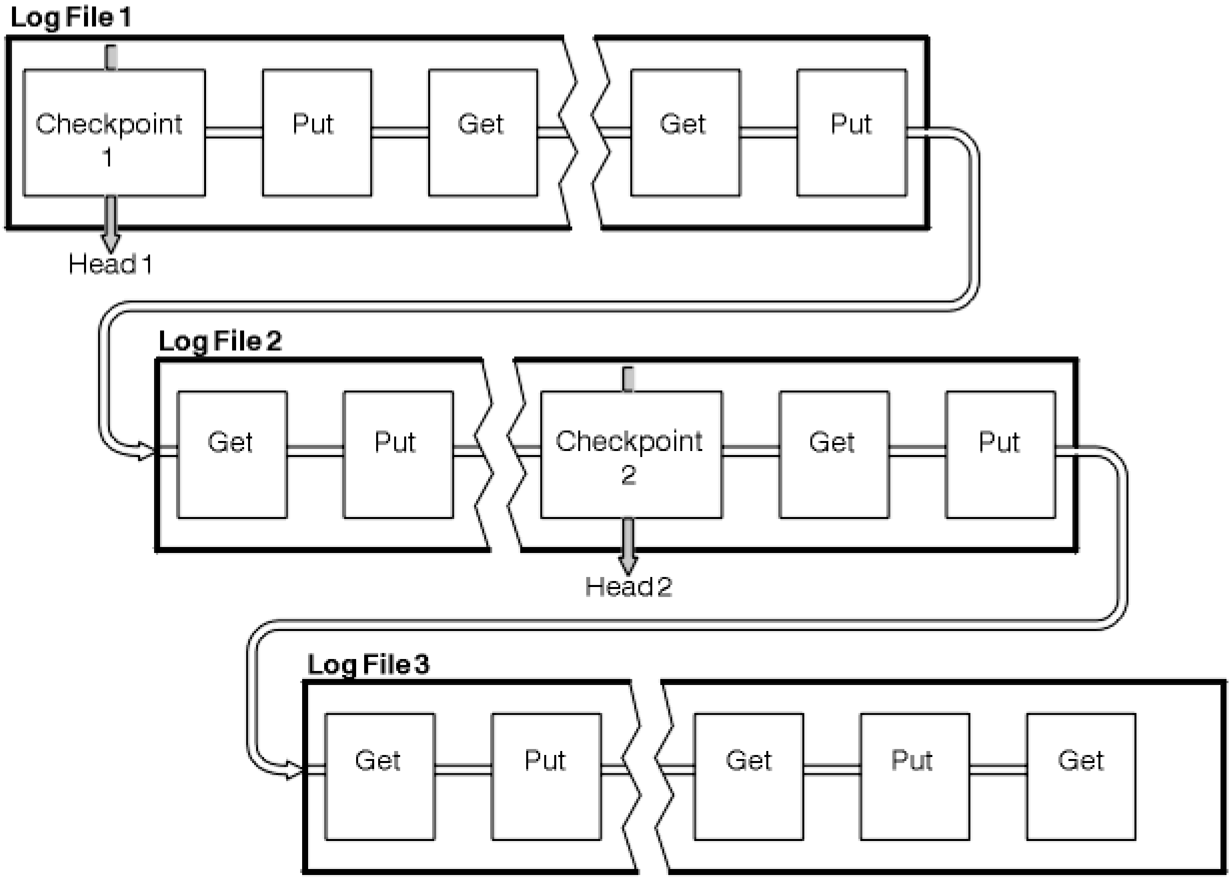

Figure 80. Checkpointing

Checkpointing with long-running transactions

How a long-running transaction affects reuse of log files.

Figure 81 on page 463 shows how a long-running transaction affects reuse of log files. In the example, a long-running transaction has made an entry to the log, shown as LR 1, after the first checkpoint shown. The transaction does not complete (at point LR 2) until after the third checkpoint. All the log information from LR 1 onwards is retained to allow recovery of that transaction, if necessary, until it has completed.

After the long-running transaction has completed, at LR 2, the head of the log logically moves to Checkpoint 3, the latest logged checkpoint. The files containing log records before Checkpoint 3, Head 2, are no longer needed. If you are using circular logging, the space can be reused.

If the primary log files are completely full before the long-running transaction completes, secondary log files might be used to avoid the logs getting full.

Activities which are entirely under the control of the queue manager, for example checkpointing, are scheduled to try and keep the activity within the primary log.

However, when secondary log space is required to support behavior outside of the control of the queue manager (for example the duration of one of your transactions) the queue manager tries using any defined secondary log space, to allow that activity to complete.

If that activity does not complete by the time 80% of the total log space is in use, the queue manager initiates action to reclaim log space, regardless of the fact that this has an impact on the application.

When the log head is moved and you are using circular logging, the primary log files might become eligible for reuse and the logger, after filling the current file, reuses the first primary file available to it. If you are using linear logging, the log head is still moved down the active pool and the first file becomes inactive. A new primary file is formatted and added to the bottom of the pool in readiness for future logging activities.

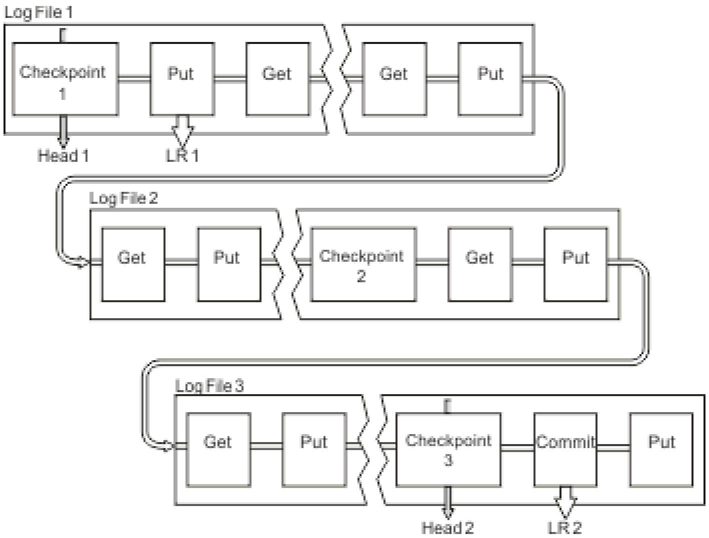

Figure 81. Checkpointing with a long-running transaction

Calculating the size of the log

Estimating the size of log a queue manager needs.

After deciding whether the queue manager uses circular or linear logging, you need to estimate the size of the Active log that the queue manager needs. The size of the active log is determined by the following log configuration parameters:

LogFilePages
The size of each primary and secondary log file in units of 4K pages

LogPrimaryFiles
The number of preallocated primary log files

LogSecondaryFiles
The number of secondary log files that can be created for use when the primary log files are becoming full

Notes:

1. You can change the number of primary and secondary log files each time the queue manager starts,although you might not notice the effect of the change you make to the secondary logs immediately.
2. You cannot change the log file size; you must determine it **before** creating the queue manager.
3. The number of primary log files and the log file size determine the amount of log space that is preallocated when the queue manager is created.
4. The total number of primary and secondary log files cannot exceed 511 on UNIX and Linux systems, or 255 on Windows, which in the presence of long-running transactions, limits the maximum amount of log space available to the queue manager for restart recovery. The amount of log space the queue manager might need for media recovery does not share this limit.
5. When *circular* logging is being used, the queue manager reuses primary and secondary log space. The queue manager will, up to a limit, allocate a secondary log file when a log file becomes full, and the next primary log file in the sequence is not available.

   See “How large should I make my active log? ” on page 464 for information on the number of logs you need to allocate. The primary log extents are used in sequence and that sequence does not change.

   For example, if you have three primary logs 0, 1, and 2, the order of use is 0,1,2 followed by 1,2,0, 2,0,1, back to 0,1,2 and so on. Any secondary logs you have allocated are interspersed as required.

6. Primary log files are made available for reuse during a checkpoint. The queue manager takes both the primary and secondary log space in to consideration before taking a checkpoint because the amount of log space is running low.

See “Log defaults for IBM MQ” on page 99 for more information.

How large should I make my active log?

Estimating the size of active log a queue manager needs.

The size of the active log is limited by:

```
logsize = (primaryfiles + secondaryfiles) * logfilepages * 4096
```

The log should be large enough to cope with your longest running transaction running when the queue manager is writing the maximum amount of data per second to disk.

If your longest running transaction runs for N seconds, and the maximum amount of data per second written to disk by the queue manager is B bytes per second in the log, your log should be at least:

```
logsize >= 2 * (N+1) * B
```

The queue manager is likely to be writing the maximum amount of data per second to disk when you are running at peak workload, or it might be when you are recording media images.

If a transaction runs for so long that the log extent containing its first log record is not contained within the active log, the queue manager rolls back active transactions one at a time, starting with the transaction with the oldest log record.

The queue manager needs to make old log extents inactive before the maximum number of primary and secondary files are being used, and the queue manager needs to allocate another log extent.

Decide how long you want your longest running transaction to run, before the queue manager is allowed to roll it back. Your longest running transaction might be waiting for slow network traffic or, in the case of a poorly designed transaction, waiting for user input.

You can investigate how long your longest running transaction runs for, by issuing the following **runmqsc** command:

```
DISPLAY CONN(*) UOWLOGDA UOWLOGTI
```

Issuing the `dspmqtrn -a` command, shows all the XA and non XA commands in all states.

Issuing this command lists the date and time that the first log record was written for all of your current transactions.

Attention: For the purposes of calculating the log size, it is the time since the first log record was written that matters, not the time since the application or transaction started. Round up the length of your longest running transaction to the nearest second. This is because of optimizations in the queue manager.

The first log record can be written long after the application started, if the application begins by, for example, issuing an MQGET call that waits for a length of time before actually getting a message.

By reviewing the maximum observed date and time output from the

```
DISPLAY CONN(*) UOWLOGDA UOWLOGTI
```

command you issued originally, from the current date and time, you can estimate how long your longest running transaction runs.

Ensure you run this **runmqsc** command repeatedly while your longest running transactions are running at peak workload so that you do not underestimate the length of your longest running transaction.

In IBM MQ 8.0 use the operating system tools, for example, **iostat** on UNIX platforms.

For example, if the operating system tools show that the logical bytes per second written to the log is approx 12 MB per second, using:

```
DISPLAY CONN(*) UOWLOGDA UOWLOGTI
```

shows that the longest running transaction was:

```

CONN(57E14F6820700069)
EXTCONN(414D51436D61726B2020202020202020)
TYPE(CONN)
APPLTAG(msginteg_r)                     UOWLOGDA(2016-09-20)
UOWLOGTI(16.44.14)
```

As the current date and time was 2016-09-20 16.44.19, this transaction had been running for 5 seconds. However, you require tolerating transactions running for 10 seconds before the queue manager rolls them back. So your log size should be:

```
2 * (10 + 1) * 12 = 264 MB
```

.

Using the default **LogFilePages** which is 4096, you need to make sure that the sum of your **LogPrimaryFiles** and **LogSecondaryFiles** is at least 17:

```
264 MB < 17 * 4096 * 4096
```

If you size your log so that your expected workload runs within the primary files:

- The secondary files provide some contingency in case additional log space is needed.
- Circular logging always using preallocated primary files, which is marginally faster than allocating and deallocating secondary files.
- The queue manager uses only the space remaining in the primary files to calculate when to take the next checkpoint.

Therefore, in the preceding example, set the following values:

- **LogFilePages** = 4096
- **LogPrimaryFiles** = 17
- **LogSecondaryFiles** = 5

Note the following:

- In this example, 5 secondaries is more than 20 per cent of the active log space.

  You should be aware that the queue manager takes action to reduce log space usage when more than 80 per cent of the total log space is in use.
- Perform the same calculation whether you are using linear or circular logging.

  It makes no difference whether you are calculating the size of a linear or circular active log, as the concept of the active log means the same in both linear logging and circular logging.
- The log extents needed for media recovery only are not within the active log, and are therefore not counted in the number of primary and secondary files.

What happens if I make my log too small?

Points you need to consider when estimating the minimum size of the log.

If you make your log too small:

- Long running transactions will get backed out.
- You might get FFDCs informing you that the log is too small.
- The next checkpoint wants to start before the previous one has ended.

Important: No matter how inaccurately you estimate the size of your log, data integrity is maintained.

See “Restart recovery” on page 459 for an explanation of checkpoints. If the amount of log space left in the active log extents is becoming short, the queue manager schedules checkpoints more frequently.

A checkpoint takes some amount of time; it is not instantaneous. The more data that needs to be recorded in the checkpoint, the longer the checkpoint takes. If the log is small checkpoints can overlap, meaning that the next checkpoint is requested before the previous checkpoint has ended. If this happens error messages are written.

If long running transactions get backed out, or checkpoints overlap, the queue manager continues processing the workload. Short-lived transactions continue running as normal.

However, the queue manager is not running optimally and performance might be degraded. You should restart the queue manager with sufficient log space.

What happens if I make my log too large?

Points you need to consider when estimating the maximum size of the log.

If you make your log too large:

- You might increase the time taken for an emergency restart, although this is unlikely.
- You are using unnecessary disk space.
- Very long running transaction are tolerated.

Important: No matter how inaccurately you estimate the size of your log, data integrity is maintained.

See “Restart recovery” on page 459 for a description of how the queue manager reads the log on restart. The queue manager replays the log from the last checkpoint, and then resolves all transactions that were active when the queue manager ended.

To resolve a transaction, the queue manager reads back all the log records associated with that transaction. These log records might predate the last checkpoint.

By allocating the queue manager a very large log, you are giving the queue manager permission to read every log record in the log on restart, although usually the queue manager does not have to do this. Potentially, in the unlikely event that this happens, that process could take a long time.

If checkpointing had unexpectedly stopped before the queue manager ended, that dramatically increases the restart time for a queue manager with a large log. Limiting the size of the log limits the emergency restart time.

To avoid these problems you should ensure that:

- Your workload can comfortably fit into a log that is not excessively large.
- You avoid long running transactions.

Managing logs

You need to manage linear logs yourself. However, circular logs are nearly self-managing, but sometimes need intervention to resolve space problems.

On circular logging, the queue manager reclaims freed space in the log files. This activity is not apparent to the user, and you do not usually see the amount of disk space used reduce, because the space allocated is quickly reused.

On linear logging, the log might fill if a checkpoint has not been taken for a long time, or if a long-running transaction wrote a log record a long time ago. The queue manager tries to take checkpoints often enough to avoid the first problem.

distributed If the log fills, message AMQ7463 is issued. In addition, if the log fills because a long-running transaction has prevented the space being released, message AMQ7465 is issued.

Of the log records, only those written since the start of the last complete checkpoint, and those written by any active transactions, are needed to restart the queue manager.

Over time, the oldest log records written become unnecessary for restarting the queue manager.

When a long-running transaction is detected, activity is scheduled to asynchronously rollback that transaction. If, for some unexpected reason, that asynchronous rollback were to fail, some MQI calls return MQRC_RESOURCE_PROBLEM in that situation.

Note that space is reserved to commit or roll back all in-flight transactions, so **MQCMIT** or **MQBACK** should not fail.

The queue manager rolls back transactions that run for a long duration. An application that has a transaction is rolled back in this way cannot perform subsequent **MQPUT** or **MQGET** operations specifying sync point under the same transaction.

However, transactions ended manually start a new log. Note, that whereas new log space is allocated immediately, log space that has been released takes a finite time to be freed up.

An attempt to put or get a message under sync point in this state returns MQRC_BACKED_OUT. The application can then issue **MQCMIT**, which returns MQRC_BACKED_OUT, or **MQBACK** and start a new transaction. When the transaction consuming too much log space has been rolled back, the log space is released and the queue manager continues to operate normally.

What happens when a disk gets full

The queue manager logging component can cope with a full disk, and with full log files. If the disk containing the log fills, the queue manager issues message AMQ6709 and an error record is taken.

The log files are created at their fixed size, rather than being extended as log records are written to them. This means that IBM MQ can run out of disk space only when it is creating a new file; it cannot run out of space when it is writing a record to the log. IBM MQ always knows how much space is available in the existing log files, and manages the space within the files accordingly.

If you fill the drive containing the log files, you might be able to free some disk space. If you are using a linear log, there might be some inactive log files in the log directory, and you can copy these files to another drive or device.

Circular logging returns a resource problem.

If you still run out of space, check that the configuration of the log in the queue manager configuration file is correct. You might be able to reduce the number of primary or secondary log files so that the log does not outgrow the available space.

You cannot alter the size of the log files for an existing queue manager. The queue manager requires that all log extents are the same size.

Managing log files

Allocate sufficient space for your log files. For linear logging, you can delete old log files when they are no longer required.

Information specific to circular logging

If you are using circular logging, ensure that there is sufficient space to hold the log files when you configure your system (see "Log defaults for IBM MQ" on page 99 and "Queue manager logs" on page 107). The amount of disk space used by the log does not increase beyond the configured size, including space for secondary files to be created when required.

Information specific to linear logging

If you are using a linear log, the log files are added continually as data is logged, and the amount of disk space used increases with time. If the rate of data being logged is high, disk space is used rapidly by new log files.

Over time, the older log files for a linear log are no longer required to restart the queue manager or to perform media recovery of any damaged objects. The following are methods for determining which log files are still required:

Logger event messages

When a significant event occurs, for example a record media image, logger event messages are generated. The contents of logger event messages specify the log files that are still required for queue manager restart, and media recovery. For more information about logger event messages, see Logger events

Queue manager status

Running the MQSC command, DISPLAY QMSTATUS, or the PCF command, Inquire Queue Manager Status, returns queue manager information, including details of the required log files. For more information about MQSC commands, see Script (MQSC) Commands, and for information about PCF commands, see Automating administration tasks.

Queue manager messages

Periodically, the queue manager issues a pair of messages to indicate which of the log files are needed:

- Message AMQ7467I gives the name of the oldest log file required to restart the queue manager. This log file and all newer log files must be available during queue manager restart.
- Message AMQ7468I gives the name of the oldest log file needed for media recovery.

To determine "older" and "newer" log files, use the log file number rather than the modification times applied by the file system.

Information applicable to both types of logging

Only log files required for queue manager restart, active log files, are required to be online. Inactive log files can be copied to an archive medium such as tape for disaster recovery, and removed from the log directory. Inactive log files that are not required for media recovery can be considered as superfluous log files. You can delete superfluous log files if they are no longer of interest to your operation.

If any log file that is needed cannot be found, operator message AMQ6767E is issued. Make the log file, and all subsequent log files, available to the queue manager and try the operation again.

Note: When performing media recovery, all the required log files must be available in the log file directory at the same time. Make sure that you take regular media images of any objects you might want to recover to avoid running out of disk space to hold all the required log files.

For example, to take a media image of all your objects in your queue manager, run the **rcdmqimg** command as shown in the following examples:

On Windows

```
rcdmqimg -m QMNAME -t all *
```

On UNIX and Linux

```
rcdmqimg -m QMNAME -t all "*"
```

Running **rcdmqimg** moves the media log sequence number (LSN) forwards. For further details on log sequence numbers, see “Dumping the contents of the log using the dmpmqlog command” on page 472. **rcdmqimg** does not run automatically, therefore must be run manually or from an automatic task you have created. For more information about this command, see rcdmqimg and dmpmqlog.

Note: Messages AMQ7467 and AMQ7468 can also be issued at the time of running the `rcdmqimg` command.

Determining superfluous log files - linear logging only
For circular logging, never delete data from the log directory. When managing linear log files, it is important to be sure which files can be deleted or archived. This information will assist you in making this decision.

Do not use the file system's modification times to determine "older" log files. Use only the log file number. The queue manager's use of log files follows complex rules, including pre-allocation and formatting of log files before they are needed. You might see log files with modification times that would be misleading if you try to use these times to determine relative age.

To determine the oldest log file needed, there are three places available to you to use:

- The DISPLAY QMSTATUS command
- Logger event messages and, finally
- Error log messages

For the DISPLAY QMSTATUS command, to determine the oldest log extent needed to:

- Restart the queue manager, issue the command `DISPLAY QMSTATUS RECLOG`.
- Perform media recovery, issue the command `DISPLAY QMSTATUS MEDIALOG`.

In general a lower log file number implies an older log. Unless you have a very high log file turnover, of the order of 3000 log files per day for 10 years, you do not need to cater for the number wrapping at 9 999 999. In this case, you can archive any log file with a number less than the RECLOG value, and you can delete any log file with a number less than both the RECLOG and MEDIALOG values.

Attention: The log file wraps, so the next number after 9 999 999 is zero.

Log file location

When choosing a location for your log files, remember that operation is severely affected if IBM MQ fails to format a new log because of lack of disk space.

If you are using a circular log, ensure that there is sufficient space on the drive for at least the configured primary log files. Also leave space for at least one secondary log file, which is needed if the log has to grow.

If you are using a linear log, allow considerably more space; the space consumed by the log increases continuously as data is logged.

You should place the log files on a separate disk drive from the queue manager data.

Data integrity on this device is paramount - you should allow for built in redundancy.

It might also be possible to place the log files on multiple disk drives in a mirrored arrangement. This protects against failure of the drive containing the log. Without mirroring, you could be forced to go back to the last backup of your IBM MQ system.

Using the log for recovery

Using logs to recover from failures.

There are several ways that your data can be damaged. IBM MQ helps you to recover from:

- A damaged data object
- A power loss in the system
- A communications failure

This section looks at how the logs are used to recover from these problems.

Recovering from power loss or communications failures

IBM MQ can recover from both communications failures and loss of power. In addition, it can sometimes recover from other types of problem, such as inadvertent deletion of a file.

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

If you lose power, when the queue manager is restarted IBM MQ restores the queues to their committed state at the time of the failure. This ensures that no persistent messages are lost. Nonpersistent messages are discarded; they do not survive when IBM MQ stops abruptly.

Recovering damaged objects

There are ways in which an IBM MQ object can become unusable, for example because of inadvertent damage. You must then recover either your complete system or some part of it. The action required depends on when the damage is detected, whether the log method selected supports media recovery, and which objects are damaged.

Media recovery

Media recovery re-creates objects from information recorded in a linear log. For example, if an object file is inadvertently deleted, or becomes unusable for some other reason, media recovery can re-create it. The information in the log required for media recovery of an object is called a *media image*.

A media image is a sequence of log records containing an image of an object from which the object itself can be re-created.

The first log record required to re-create an object is known as its *media recovery record* ; it is the start of the latest media image for the object. The media recovery record of each object is one of the pieces of information recorded during a checkpoint.

When an object is re-created from its media image, it is also necessary to replay any log records describing updates performed on the object since the last image was taken.

Consider, for example, a local queue that has an image of the queue object taken before a persistent message is put onto the queue. In order to re-create the latest image of the object, it is necessary to replay the log entries recording the putting of the message to the queue, in addition to replaying the image itself.

When an object is created, the log records written contain enough information to completely re-create the object. These records make up the first media image of the object. Then, at each shutdown, the queue manager records media images automatically as follows:

- Images of all process objects and queues that are not local
- Images of empty local queues

Media images can also be recorded manually using the **rcdmqimg** command, described in rcdmqimg. This command writes a media image of the IBM MQ object.

When a media image has been written, only the logs that hold the media image, and all the logs created after this time, are required to re-create damaged objects. The benefit of creating media images depends on such factors as the amount of free storage available, and the speed at which log files are created.

Recovering from media images

A queue manager automatically recovers some objects from their media image during startup of the queue manager. It recovers a queue automatically if it was involved in any transaction that was incomplete when the queue manager last shut down, and is found to be corrupted or damaged during the restart processing.

You must recover other objects manually, using the **rcrmqobj** command, which replays the records in the log to re-create the IBM MQ object. The object is re-created from its latest image found in the log, together with all applicable log events between the time the image was saved and the time the re-create command was issued. If an IBM MQ object becomes damaged, the only valid actions that can be performed are either to delete it or to re-create it by this method. Nonpersistent messages cannot be recovered in this way.

See rcrmqobj for further details of the **`rcrmqobj`** command.

The log file containing the media recovery record, and all subsequent log files, must be available in the log file directory when attempting media recovery of an object. If a required file cannot be found, operator message AMQ6767 is issued and the media recovery operation fails. If you do not take regular media images of the objects that you want to re-create, you might have insufficient disk space to hold all the log files required to re-create an object.

Recovering damaged objects during startup

If the queue manager discovers a damaged object during startup, the action it takes depends on the type of object and whether the queue manager is configured to support media recovery.

If the queue manager object is damaged, the queue manager cannot start unless it can recover the object. If the queue manager is configured with a linear log, and thus supports media recovery, IBM MQ automatically tries to re-create the queue manager object from its media images. If the log method selected does not support media recovery, you can either restore a backup of the queue manager or delete the queue manager.

If any transactions were active when the queue manager stopped, the local queues containing the persistent, uncommitted messages put or got inside these transactions are also required to start the queue manager successfully. If any of these local queues is found to be damaged, and the queue manager supports media recovery, it automatically tries to re-create them from their media images. If any of the queues cannot be recovered, IBM MQ cannot start.

If any damaged local queues containing uncommitted messages are discovered during startup processing on a queue manager that does not support media recovery, the queues are marked as damaged objects and the uncommitted messages on them are ignored. This situation is because it is not possible to perform media recovery of damaged objects on such a queue manager and the only action left is to delete them. Message AMQ7472 is issued to report any damage.

Recovering damaged objects at other times

Media recovery of objects is automatic only during startup. At other times, when object damage is detected, operator message AMQ7472 is issued and most operations using the object fail. If the queue manager object is damaged at any time after the queue manager has started, the queue manager performs a pre-emptive shutdown. When an object has been damaged you can delete it or, if the queue manager is using a linear log, attempt to recover it from its media image using the `rcrmqobj` command (see rcrmqobj for further details).

Protecting IBM MQ log files

Do not touch the log files when a queue manager is running, recovery might be impossible. Use super user or mqm authority to protect log files against inadvertent modification.

Do not remove the active log files manually when an IBM MQ queue manager is running. If a user inadvertently deletes the log files that a queue manager needs to restart, IBM MQ **does not** issue any errors and continues to process data *including persistent messages*. The queue manager shuts down normally, but can fail to restart. Recovery of messages then becomes impossible.

Users with the authority to remove logs that are being used by an active queue manager also have authority to delete other important queue manager resources (such as queue files, the object catalog, and IBM MQ executable files). They can therefore damage, perhaps through inexperience, a running or dormant queue manager in a way against which IBM MQ cannot protect itself.

Exercise caution when conferring super user or mqm authority.

Dumping the contents of the log using the dmpmqlog command

How to use the `dmpmqlog` command to dump the contents of the queue manager log.

Use the `dmpmqlog` command to dump the contents of the queue manager log. By default all active log records are dumped, that is, the command starts dumping from the head of the log (usually the start of the last completed checkpoint).

The log can usually be dumped only when the queue manager is not running. Because the queue manager takes a checkpoint during shutdown, the active portion of the log usually contains a small number of log records. However, you can use the `dmpmqlog` command to dump more log records using one of the following options to change the start position of the dump:

- Start dumping from the *base* of the log. The base of the log is the first log record in the log file that contains the head of the log. The amount of additional data dumped in this case depends on where the head of the log is positioned in the log file. If it is near the start of the log file, only a small amount of additional data is dumped. If the head is near the end of the log file, significantly more data is dumped.
- Specify the start position of the dump as an individual log record. Each log record is identified by a unique *log sequence number (LSN)*. In the case of circular logging, this starting log record cannot be before the base of the log; this restriction does not apply to linear logs. You might need to reinstate inactive log files before running the command. You must specify a valid LSN, taken from previous `dmpmqlog` output, as the start position.

  For example, with linear logging you can specify the `nextlsn` from your last `dmpmqlog` output. The `nextlsn` appears in `Log File Header` and indicates the LSN of the next log record to be written. Use this as a start position to format all log records written since the last time the log was dumped.
- **For linear logs only**, you can instruct `dmpmqlog` to start formatting log records from any given log file extent. In this case, `dmpmqlog` expects to find this log file, and each successive one, in the same directory as the active log files. This option does not apply to circular logs, where `dmpmqlog` cannot access log records prior to the base of the log.

The output from the `dmpmqlog` command is the `Log File Header` and a series of formatted log records. The queue manager uses several log records to record changes to its data.

Some of the information that is formatted is only of use internally. The following list includes the most useful log records:

Log File Header

Each log has a single log file header, which is always the first thing formatted by the `dmpmqlog` command. It contains the following fields:

logactive	The number of primary log extents.
loginactive	The number of secondary log extents.
logsize	The number of 4 KB pages per extent.
baselsn	The first LSN in the log extent containing the head of the log.
nextlsn	The LSN of the next log record to be written.
headlsn	The LSN of the log record at the head of the log.
tailsn	The LSN identifying the tail position of the log.
hflag1	Whether the log is CIRCULAR or LOG RETAIN (linear).
HeadExtentID	The log extent containing the head of the log.

Log Record Header

Each log record within the log has a fixed header containing the following information:

LSN	The log sequence number.
LogRecdType	The type of the log record.

XTranid — The transaction identifier associated with this log record (if any).

A *TranType* of MQI indicates an IBM MQ-only transaction. A *TranType* of XA is involved with other resource managers. Updates involved within the same unit of work have the same *XTranid*.

QueueName — The queue associated with this log record (if any).

Qid — The unique internal identifier for the queue.

PrevLSN — The LSN of the previous log record within the same transaction (if any).

Start Queue Manager
This logs that the queue manager has started.

StartDate — The date that the queue manager started.

StartTime — The time that the queue manager started.

Stop Queue Manager
This logs that the queue manager has stopped.

StopDate — The date that the queue manager stopped.

StopTime — The time that the queue manager stopped.

ForceFlag — The type of shutdown used.

Start Checkpoint
This denotes the start of a queue manager checkpoint.

End Checkpoint
This denotes the end of a queue manager checkpoint.

ChkPtLSN — The LSN of the log record that started this checkpoint.

Put Message
This logs a persistent message put to a queue. If the message was put under sync point, the log record header contains a non-null *XTranid*. The remainder of the record contains:

MapIndex — An identifier for the message on the queue. It can be used to match the corresponding `MQGET` that was used to get this message from the queue. In this case a subsequent *Get Message* log record can be found containing the same `QueueName` and `MapIndex`. At this point the `MapIndex` identifier can be reused for a subsequent put message to that queue.

Data — Contained in the hex dump for this log record is various internal data followed by a representation of the Message Descriptor (eyecatcher MD) and then the message data itself.

Put Part
Persistent messages that are too large for a single log record are logged as multiple *Put Part* log records followed by a single *Put Message* record. If there are *Put Part* records, then the *PrevLSN* field will chain the *Put Part* records and the final *Put Message* record together.

Data — Continues the message data where the previous log record left off.

Get Message
Only gets of persistent messages are logged. If the message was got under sync point, the log record header contains a non-null *XTranid*. The remainder of the record contains:

MapIndex	Identifies the message that was retrieved from the queue. The most recent *Put Message* log record containing the same *QueueName* and *MapIndex* identifies the message that was retrieved.
QPriority	The priority of the message retrieved from the queue.

Start Transaction
Indicates the start of a new transaction. A TranType of MQI indicates an IBM MQ-only transaction. A TranType of XA indicates one that involves other resource managers. All updates made by this transaction will have the same *XTranid*.

Prepare Transaction
Indicates that the queue manager is prepared to commit the updates associated with the specified *XTranid*. This log record is written as part of a two-phase commit involving other resource managers.

Commit Transaction
Indicates that the queue manager has committed all updates made by a transaction.

Rollback Transaction
This denotes the queue manager's intention to roll back a transaction.

End Transaction
This denotes the end of a rolled-back transaction.

Transaction Table
This record is written during sync point. It records the state of each transaction that has made persistent updates. For each transaction the following information is recorded:

XTranid	The transaction identifier.
FirstLSN	The LSN of the first log record associated with the transaction.
LastLSN	The LSN of the last log record associated with the transaction.

Transaction Participants
This log record is written by the XA Transaction Manager component of the queue manager. It records the external resource managers that are participating in transactions. For each participant the following is recorded:

RMName	The name of the resource manager.
RMID	The resource manager identifier. This is also logged in subsequent *Transaction Prepared* log records that record global transactions in which the resource manager is participating.
SwitchFile	The switch load file for this resource manager.
XAOpenString	The XA open string for this resource manager.
XACloseString	The XA close string for this resource manager.

Transaction Prepared
This log record is written by the XA Transaction Manager component of the queue manager. It indicates that the specified global transaction has been successfully prepared. Each of the participating resource managers will be instructed to commit. The *RMID* of each prepared resource manager is recorded in the log record. If the queue manager itself is participating in the transaction a *Participant Entry* with an *RMID* of zero will be present.

Transaction Forget
This log record is written by the XA Transaction Manager component of the queue manager. It follows the *Transaction Prepared* log record when the commit decision has been delivered to each participant.

Purge Queue
This logs the fact that all messages on a queue have been purged, for example, using the MQSC command CLEAR QUEUE.

Queue Attributes
This logs the initialization or change of the attributes of a queue.

Create Object
This logs the creation of an IBM MQ object.

ObjName	The name of the object that was created.
UserId	The user ID performing the creation.

Delete Object
This logs the deletion of an IBM MQ object.

ObjName	The name of the object that was deleted.

Backing up and restoring IBM MQ queue manager data

Backing up queue managers and queue manager data.

Periodically, you can take measures to protect queue managers against possible corruption caused by hardware failures. There are three ways of protecting a queue manager:

Back up the queue manager data
If the hardware fails, a queue manager might be forced to stop. If any queue manager log data is lost due to the hardware failure, the queue manager might be unable to restart. If you back up queue manager data you might be able to recover some, or all, of the lost queue manager data.

In general, the more frequently you back up queue manager data, the less data you lose in the event of hardware failure that results in loss of integrity of the recovery log.

To back up queue manager data, the queue manager must not be running.

To back up and restore queue manager data see:

- "Backing up queue manager data" on page 476.
- "Restoring queue manager data" on page 476.

Use a backup queue manager
If the hardware failure is severe, a queue manager might be unrecoverable. In this situation, if the unrecoverable queue manager has a dedicated backup queue manager, the backup queue manager can be activated in place of the unrecoverable queue manager. If it was updated regularly, the backup queue manager log can contain log data that includes the last complete log from the unrecoverable queue manager.

A backup queue manager can be updated while the existing queue manager is still running.

To create and activate a backup queue manager see:

- "Creating a backup queue manager" on page 477.
- "Starting a backup queue manager" on page 478.

Back up the queue manager configuration only
If the hardware fails, a queue manager might be forced to stop. If both the queue manager configuration and log data is lost due to the hardware failure, the queue manager will be unable to restart or to be recovered from the log. If you back up the queue manager configuration you would be able to recreate the queue manager and all of its objects from saved definitions.

To back up queue manager configuration, the queue manager must be running.

To back up and restore the queue manager configuration see:

- "Backing up queue manager configuration" on page 479
- "Restoring queue manager configuration" on page 479

Backing up queue manager data

Backing up queue manager data can help you to guard against possible loss of data caused by hardware errors.

Before you begin

Ensure that the queue manager is not running. If you try to take a backup of a running queue manager, the backup might not be consistent because of updates in progress when the files were copied. If possible, stop your queue manager by running the `endmqm -w` command (a wait shutdown), only if that fails, use the `endmqm -i` command (an immediate shutdown).

About this task

To take a backup copy of a queue manager's data, complete the following tasks:

1. Search for the directories under which the queue manager places its data and its log files, by using the information in the configuration files. For more information, see "Changing IBM MQ and queue manager configuration information" on page 80.

   Note: The names that appear in the directory are transformed to ensure that they are compatible with the platform on which you are using IBM MQ. For more information about name transformations, see Understanding IBM MQ file names.
2. Take copies of all the queue manager's data and log file directories, including all subdirectories.

   Make sure that you do not miss any files, especially the log control file, as described in "What logs look like" on page 457, and the configuration files as described in "Initialization and configuration files" on page 170. Some of the directories might be empty, but you need them all to restore the backup at a later date.
3. Preserve the ownerships of the files. For IBM MQ for UNIX and Linux systems, you can do this with the `tar` command. (If you have queues larger than 2 GB, you cannot use the `tar` command. For more information, see Enabling large queues.

   Note: When you upgrade to IBM WebSphere MQ 7.5 and later, ensure to take a backup of the **`.ini`** file and the registry entries. The queue manager information is stored in the **`.ini`** file and can be used to revert to a previous version of IBM MQ.

Restoring queue manager data

Follow these steps to restore a backup of a queue manager's data.

Before you begin

Ensure that the queue manager is not running.

When restoring a backup of a queue manager in a cluster, see "Recovering a cluster queue manager" on page 292 and Clustering: Availability, multi-instance, and disaster recovery for more information.

About this task

To restore a backup of a queue manager's data:

1. Find the directories under which the queue manager places its data and its log files, by using the information in the configuration files.
2. Empty the directories into which you are going to place the backed-up data.
3. Copy the backed-up queue manager data and log files into the correct places.
4. Update the configuration information files.

Check the resulting directory structure to ensure that you have all the required directories.

For more information about IBM MQ directories and subdirectories, see Directory structure on Windows systems and Directory content on UNIX and Linux systems.

Make sure that you have a log control file as well as the log files. Also check that the IBM MQ and queue manager configuration files are consistent so that IBM MQ can look for the restored data in the correct places.

For circular logging, back up the queue manager data and log file directories at the same time so that you can restore a consistent set of queue manager data and logs.

For linear logging, back up the queue manager data and log file directories at the same time. It is possible to restore only the queue manager data files if a corresponding complete sequence of log files is available.

Note: When you upgrade to IBM WebSphere MQ 7.5 and later, ensure to take a backup of the **`.ini`** file and the registry entries. The queue manager information is stored in the **`.ini`** file and can be used to revert to a previous version of IBM MQ.

Results

If the data was backed up and restored correctly, the queue manager will now start.

Using a backup queue manager

An existing queue manager can have a dedicated backup queue manager.

A backup queue manager is an inactive copy of the existing queue manager. If the existing queue manager becomes unrecoverable due to severe hardware failure, the backup queue manager can be brought online to replace the unrecoverable queue manager.

The existing queue manager log files must regularly be copied to the backup queue manager to ensure that the backup queue manager remains an effective method for disaster recovery. The existing queue manager does not need to be stopped for log files to be copied, however you should only copy a log file if the queue manager has finished writing to it; see "Updating a backup queue manager" on page 478 for information on how to ensure a specific log file is not being written to anymore, so that it can be safely copied.

Note: Because the existing queue manager log is continually updated, there is always a slight discrepancy between the existing queue manager log and the log data copied to the backup queue manager log. Regular updates to the backup queue manager minimizes the discrepancy between the two logs.

If a backup queue manager is required to be brought online it must be activated, and then started. The requirement to activate a backup queue manager before it is started is a preventive measure to protect against a backup queue manager being started accidentally. After a backup queue manager is activated it can no longer be updated.

For information on using a backup queue manager, see the following topics:

- "Creating a backup queue manager" on page 477
- "Updating a backup queue manager" on page 478
- "Starting a backup queue manager" on page 478

Creating a backup queue manager

You can only use a backup queue manager when using linear logging.

To create a backup queue manager for an existing queue manager, do the following:

1. Create a backup queue manager for the existing queue manager using the control command `crtmqm`. The backup queue manager requires the following:
   - To have the same attributes as the existing queue manager, for example the queue manager name, the logging type, and the log file size.
   - To be on the same platform as the existing queue manager.

- To be at an equal, or higher, code level than the existing queue manager.

2. Take copies of all the existing queue manager's data and log file directories, including all subdirectories, as described in "Backing up queue manager data" on page 476.
3. Overwrite the backup queue manager's data and log file directories, including all subdirectories, with the copies taken from the existing queue manager.
4. Execute the following control command on the backup queue manager:

```
strmqm -r BackupQMName
```

This flags the queue manager as a backup queue manager within IBM MQ, and replays all the copied log extents to bring the backup queue manager in step with the existing queue manager.

Updating a backup queue manager

To ensure that a backup queue manager remains an effective method for disaster recovery it must be updated regularly.

Regular updating lessens the discrepancy between the backup queue manager log, and the current queue manager log. There is no need to stop the queue manager to be backed up.

To update a backup queue manager, do the following:

1. Issue the following Script (MQSC) command on the queue manager to be backed up:

```
RESET QMGR TYPE(ADVANCELOG)
```

This stops any writing to the current log, and then advances the queue manager logging to the next log extent. This ensures you back up all information logged up to the current time.

2. Obtain the (new) current active log extent number by issuing the following Script (MQSC) command on the queue manager to be backed up:

```
DIS QMSTATUS CURRLOG
```

3. Copy the updated log extent files from the current queue manager log directory to the backup queue manager log directory - copy all the log extents since the last update, and up to (but not including) the current extent noted in step 2. Copy only log extent files, the ones beginning with "S...".
4. Issue the following control command on the backup queue manager:

```
strmqm -r BackupQMName
```

This replays all the copied log extents and brings the backup queue manager into step with the queue manager. When the replay finishes you receive a message that identifies all the log extents required for restart recovery, and all the log extents required for media recovery.

Warning: If you copy a `non-contiguous` set of logs to the backup queue manager log directory, only the logs up to the point where the first missing log is found will be replayed.

Starting a backup queue manager

You can substitute a backup queue manager for an unrecoverable queue manager.

To do this, perform the following steps:

1. Execute the following control command to activate the backup queue manager:

```
strmqm -a BackupQMName
```

The backup queue manager is activated. Now active, the backup queue manager can no longer be updated.

2. Execute the following control command to start the backup queue manager:

   ```
   strmqm BackupQMName
   ```

   IBM MQ regards this as restart recovery, and utilizes the log from the backup queue manager. During the last update to the backup queue manager replay will have occurred, therefore only the active transactions from the last recorded checkpoint are rolled back.

   When an unrecoverable queue manager is substituted for a backup queue manager some of the queue manager data from the unrecoverable queue manager can be lost. The amount of lost data is dependent on how recently the backup queue manager was last updated. The more recently the last update, the less queue manager data loss.

3. Restart all channels.

Check the resulting directory structure to ensure that you have all the required directories.

See Planning file system support for more information about IBM MQ directories and subdirectories.

Make sure that you have a log control file as well as the log files. Also check that the IBM MQ and queue manager configuration files are consistent so that IBM MQ can look in the correct places for the restored data.

If the data was backed up and restored correctly, the queue manager will now start.

Note: Even though the queue manager data and log files are held in different directories, back up and restore the directories at the same time. If the queue manager data and log files have different ages, the queue manager is not in a valid state and will probably not start. If it does start, your data is likely to be corrupt.

Backing up queue manager configuration

Backing up queue manager configuration can help you to rebuild a queue manager from its definitions.

To take a backup copy of a queue manager's configuration:

1. Ensure that the queue manager is running.
2. a. On AIX, HP-UX, Linux, Solaris, or Windows: Execute the Dump MQ Configuration command (dmpmqcfg) using the default formatting option of (-f mqsc) MQSC and all attributes (-a), use standard output redirection to store the definitions into a file, for example:

      ```
      dmpmqcfg -m MYQMGR -a > /mq/backups/MYQMGR.mqsc
      ```

   b. IBM i On IBM i: Execute the Dump MQ Configuration command (DMPMQMCFG) using the default formatting option of OUTPUT(*MQSC) and EXPATTR(*ALL), use the TOFILE and TOMBR to store the definitions into a physical file member, for example:

      ```
      DMPMQMCFG MQMNAME(MYQMGR) OUTPUT(*MQSC) EXPATTR(*ALL) TOFILE(QMQMSAMP/QMQSC)
      TOMBR(MYQMGRDEF)
      ```

Restoring queue manager configuration

Follow these steps to restore a backup of a queue manager's configuration.

To restore a backup of a queue manager's configuration:

1. Ensure that the queue manager is running. Note that the queue manager may have been recreated if damage to the data and logs is unrecoverable by other means.
2. Depending on your platform, execute one of the following commands:

a. Windows Linux UNIX On AIX, HP-UX, Linux, Solaris, or Windows: Execute **runmqsc** against the queue manager, use standard input redirection to restore the definitions from a script file generated by the Dump MQ Configuration (**dmpmqcfg**) command, for example:

```
runmqsc MYQMGR < /mq/backups/MYQMGR.mqsc
```

b. IBM i On IBM i: Execute **STRMQMMQSC** against the queue manager, use **SRCMBR** and **SRCFILE** to restore the definitions from the physical file member generated by the Dump MQ Configuration (**DMPMQMCFG**) command, for example:

```
STRMQMMQSC MQMNAME(MYQMGR) SRCFILE(QMQMSAMP/QMQSC) SRCMBR(MYQMGR)
```

Related reference

dmpmqcfg (dump queue manager configuration)

Configuring JMS resources

One of the ways in which a JMS application can create and configure the resources that it needs to connect to IBM MQ and access destinations for sending or receiving messages is by using the Java Naming and Directory Interface (JNDI) to retrieve administered objects from a location within the naming and directory service that is called the JNDI namespace. Before a JMS application can retrieve administered objects from a JNDI namespace, you must first create and configure the administered objects.

About this task

You can create and configure administered objects in IBM MQ by using either of the following tools:

MQ Explorer
You can use MQ Explorer to create and administer JMS object definitions that are stored in LDAP, in a local file system, or other locations.

IBM MQ JMS administration tool
The IBM MQ JMS administration tool is a command-line tool that you can use to create and configure IBM MQ JMS objects that are stored in LDAP, in a local file system, or other locations. The JMS administration tool uses a syntax that is similar to **runmqsc**, and also supports scripting.

The administration tool uses a configuration file to set the values of certain properties. A sample configuration file is supplied, which you can edit to suit your system before you start by using the tool to configure JMS resources. For more information about the configuration file, see "Configuring the JMS administration tool" on page 486.

IBM MQ JMS applications that are deployed to WebSphere Application Server need to access JMS objects from the application server JNDI repository. Therefore, if you use JMS messaging between WebSphere Application Server and IBM MQ, you must create objects in WebSphere Application Server that correspond to the objects that you create in IBM MQ.

MQ Explorer and the IBM MQ JMS administration tool cannot be used to administer IBM MQ JMS objects that are stored in WebSphere Application Server. Instead, you can create and configure administered objects in WebSphere Application Server by using either of the following tools:

WebSphere Application Server administrative console
The WebSphere Application Server administrative console is a web-based tool that you can use to manage IBM MQ JMS objects in WebSphere Application Server.

WebSphere Application Server wsadmin scripting client
The WebSphere Application Server wsadmin scripting client provides specialized commands to administer IBM MQ JMS objects in WebSphere Application Server.

If you want to use a JMS application to access the resources of an IBM MQ queue manager from within WebSphere Application Server, you must use the IBM MQ messaging provider in WebSphere Application Server, which contains a version of the IBM MQ classes for JMS. The IBM MQ resource adapter that is supplied with WebSphere Application Server is used by all applications that carry out JMS messaging with the IBM MQ messaging provider. The IBM MQ resource adapter is usually updated automatically when you apply WebSphere Application Server fix packs, but if you have previously manually updated the resource adapter, you must manually update your configuration to ensure that maintenance is applied correctly.

Related tasks

Writing IBM MQ classes for JMS applications

Related reference

runmqsc

Configuring connection factories and destinations in a JNDI namespace

JMS applications access administered objects in the naming and directory service through the Java Naming and Directory Interface (JNDI). The JMS administered objects are stored in a location within the naming and directory service that is referred to as the JNDI namespace. A JMS application can look up the administered objects to connect to IBM MQ and access destinations for sending or receiving messages.

About this task

JMS applications look up the names of the JMS objects in the naming and directory service by using contexts:

Initial context
: The initial context defines the root of the JNDI namespace. For each location in the naming and directory service, you need to specify an initial context to give a starting point from which a JMS application can resolve the names of the administered objects in that location of the naming and directory service.

Subcontexts
: A context can have one or more subcontexts. A subcontext is a subdivision of a JNDI namespace and can contain administered objects such as connection factories and destinations as well as other subcontexts. A subcontext is not an object in its own right; it is merely an extension of the naming convention for the objects in the subcontext.

You can create contexts using either MQ Explorer or the IBM MQ JMS administration tool.

Before an IBM MQ classes for JMS application can retrieve administered objects from a JNDI namespace, you must first create the administered objects using either MQ Explorer or the IBM MQ JMS administration tool. You can create and configure the following types of JMS object:

Connection factory
: A JMS connection factory object defines a set of standard configuration properties for connections. A JMS application uses a connection factory to create a connection to IBM MQ. You can create a connection factory that is specific to one of the two messaging domains, the point-to-point messaging domain and the publish/subscribe messaging domain. Alternatively, from JMS 1.1, you can create domain-independent connection factories that can be used for both point-to-point and publish/subscribe messaging.

Destination
: A JMS destination is an object that represents the target of messages that the client produces and the source of messages that a JMS application consumes. The JMS application can either use a single destination object to put messages on and to get messages from, or the application can use separate destination objects. There are two types of destination object:

- JMS queue destination used in point-to-point messaging
- JMS topic destination used in publish/subscribe messaging

The following diagram shows an example of JMS objects created in an IBM MQ JNDI namespace.

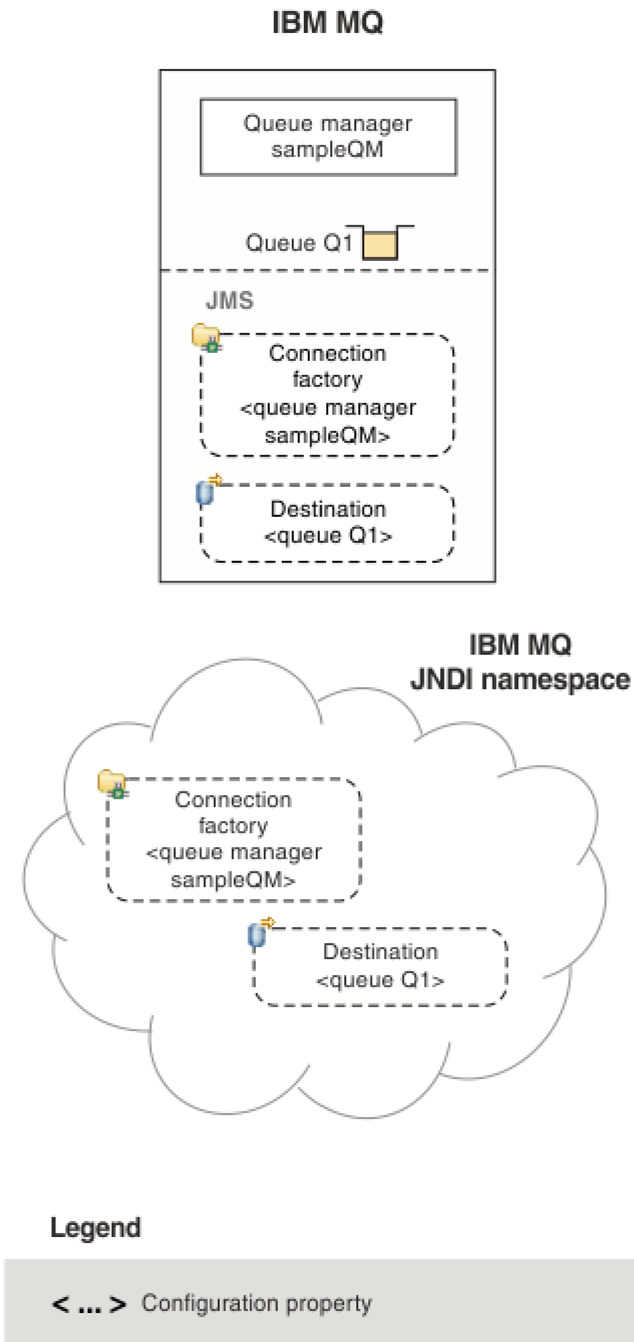

Figure 82. JMS objects created in IBM MQ

If you use JMS messaging between WebSphere Application Server and IBM MQ, you must create corresponding objects in WebSphere Application Server to use to communicate with IBM MQ. When you create one of these objects in WebSphere Application Server, it is stored in the WebSphere Application Server JNDI namespace as shown in the following diagram.

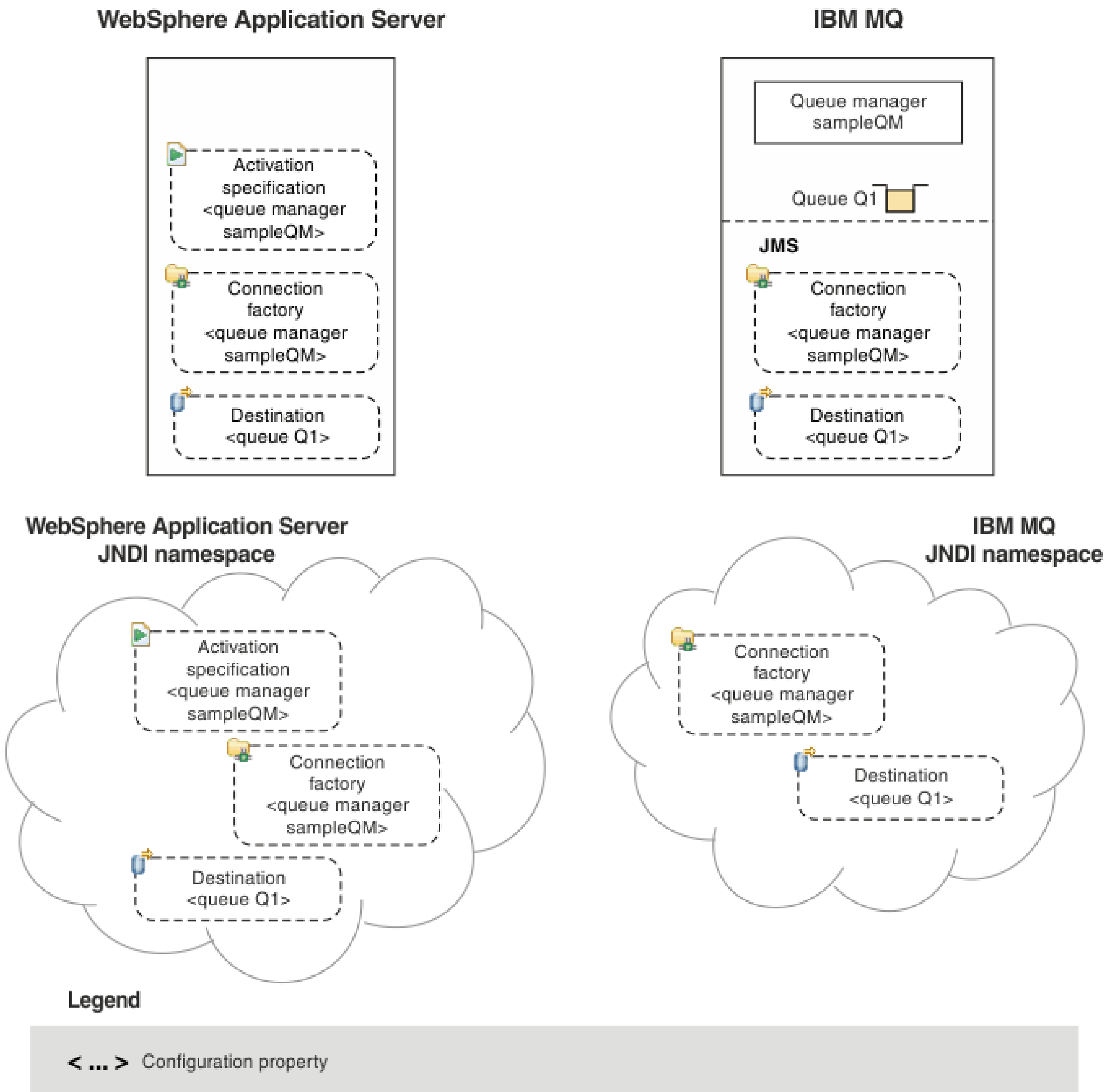

Figure 83. Objects created in WebSphere Application Server, and the corresponding objects in IBM MQ

If your application uses a message-driven bean (MDB), the connection factory is used for outbound messages only and inbound messages are received by an activation specification. Activation specifications are part of the Java EE Connector Architecture 1.5 (JCA 1.5) standard. JCA 1.5 provides a standard way to integrate JMS providers, such as IBM MQ, with Java EE application servers such as WebSphere Application Server. A JMS activation specification can be associated with one or more message driven beans (MDBs) and provides the configuration necessary for these MDBs to listen for messages arriving at a destination.

You can use either the WebSphere Application Server administrative console or wsadmin scripting commands to create and configure the JMS resources that you need.

Procedure

- To configure JMS objects for IBM MQ using MQ Explorer, see “Configuring JMS objects using MQ Explorer” on page 484.

- To configure JMS objects for IBM MQ using the IBM MQ JMS administration tool, see "Configuring JMS objects using the administration tool" on page 485.
- To configure JMS objects for WebSphere Application Server, see "Configuring JMS resources in WebSphere Application Server" on page 494.

Results

An IBM MQ classes for JMS application can retrieve the administered objects from the JNDI namespace and, if required, set or change one or more of its properties by using either the IBM JMS extensions or the IBM MQ JMS extensions.

Related tasks

Using JNDI to retrieve administered objects in a JMS application

Creating and configuring connection factories and destinations in an IBM MQ classes for JMS application

Configuring JMS objects using MQ Explorer

Use the MQ Explorer graphical user interface to create JMS objects from IBM MQ objects, and IBM MQ objects from JMS objects, as well as for administering and monitoring other IBM MQ objects.

About this task

MQ Explorer is the graphical user interface in which you can administer and monitor IBM MQ objects, whether they are hosted by your local computer or on a remote system. MQ Explorer runs on Windows and Linux x86-64. It can remotely connect to queue managers that are running on any supported platform including z/OS, enabling your entire messaging backbone to be viewed, explored, and altered from the console.

In MQ Explorer, all connection factories are stored in Connection Factories folders in the appropriate context and subcontexts.

You can perform the following types of task with MQ Explorer, either contextually from an existing object in the MQ Explorer, or from within a create new object wizard:

- Create a JMS Connection Factory from any of the following IBM MQ objects:
  - An IBM MQ queue manager, whether on your local computer or on a remote system.
  - An IBM MQ channel.
  - An IBM MQ listener.
- Add an IBM MQ queue manager to MQ Explorer using a JMS Connection Factory.
- Create a JMS queue from an IBM MQ queue.
- Create an IBM MQ queue from a JMS queue.
- Create a JMS topic from an IBM MQ topic, which can be an IBM MQ object or a dynamic topic.
- Create an IBM MQ topic from a JMS topic.

Procedure

- Start MQ Explorer, if it is not already running.

  If MQ Explorer is running and displaying the Welcome page, close the Welcome page to start administering IBM MQ objects.
- If you have not already done so, create an initial context defining the root of the JNDI namespace in which the JMS objects are stored in the naming and directory service.

  When you have added the initial context to MQ Explorer, you can create connection factory objects, destination objects, and subcontexts in the JNDI namespace.

  The initial context is displayed in the Navigator view in the JMS Administered Objects folder. Note that although the full contents of the JNDI namespace are displayed, in MQ Explorer you can edit only the IBM MQ classes for JMS objects that are stored there. For more information, see Adding an initial context.

- Create and configure the subcontexts and JMS administered objects that you need.

  For more information, see Creating and configuring JMS administered objects.
- Configure IBM MQ.

  For more information, see Configuring IBM MQ using MQ Explorer .

Related concepts

Introduction to MQ Explorer

Related tasks

Creating and configuring connection factories and destinations in an IBM MQ classes for JMS application

Configuring JMS objects using the administration tool

You can use the IBM MQ JMS administration tool to define the properties of eight types of IBM MQ classes for JMS object and to store them within a JNDI namespace. Applications can then use JNDI to retrieve these administered objects from the namespace.

About this task

The following table shows the eight types of administered objects that you can create, configure and manipulate using verbs. The Keyword column shows the strings that you can substitute for *TYPE* in the commands shown in Table 31 on page 485.

Table 31. The JMS object types that are handled by the administration tool

Object Type	Keyword	Description
MQConnectionFactory	CF	The IBM MQ implementation of the JMS ConnectionFactory interface. This represents a factory object for creating connections in both the point-to-point and publish/subscribe domains.
MQQueueConnectionFactory	QCF	The IBM MQ implementation of the JMS QueueConnectionFactory interface. This represents a factory object for creating connections in the point-to-point domain.
MQTopicConnectionFactory	TCF	The IBM MQ implementation of the JMS TopicConnectionFactory interface. This represents a factory object for creating connections in the publish/subscribe domain.
MQQueue	Q	The IBM MQ implementation of the JMS Queue interface. This represents a destination for messages in the point-to-point domain.
MQTopic	T	The IBM MQ implementation of the JMS Topic interface. This represents a destination for messages in the publish/subscribe domain.

Table 31. The JMS object types that are handled by the administration tool (continued)		
Object Type	**Keyword**	**Description**
MQXAConnectionFactory "1" on page 486	XACF	The IBM MQ implementation of the JMS XAConnectionFactory interface. This represents a factory object for creating connections in both the point-to-point and publish/subscribe domains, and where the connections use the XA versions of JMS classes.
MQXAQueueConnectionFactory "1" on page 486	XAQCF	The IBM MQ implementation of the JMS XAQueueConnectionFactory interface. This represents a factory object for creating connections in the point-to-point domain that use the XA versions of JMS classes.
MQXATopicConnectionFactory "1" on page 486	XATCF	The IBM MQ implementation of the JMS XATopicConnectionFactory interface. This represents a factory object for creating connections in the publish/subscribe domain that use the XA versions of JMS classes.

Note:

1. These classes are provided for use by vendors of application servers. They are unlikely to be directly useful to application programmers.

For more information about how to configure these objects, see "Configuring JMS objects" on page 493.

The property types and values that you need to use this tool are listed in Properties of IBM MQ classes for JMS objects.

You can also use the tool to manipulate directory namespace subcontexts within the JNDI as described in "Configuring subcontexts" on page 490.

You can also create and configure JMS administered objects with MQ Explorer.

Related tasks

Creating and configuring connection factories and destinations in an IBM MQ classes for JMS application

Using JNDI to retrieve administered objects in a JMS application

Configuring the JMS administration tool

The IBM MQ JMS administration tool uses a configuration file to set the values of certain properties. A sample configuration file is supplied, which you can edit to suit your system.

About this task

The configuration file is a plain-text file that consists of a set of key-value pairs, separated by the equal sign (=). You configure the administration tool by setting values for the three properties defined in the configuration file. The following example shows these three properties:

```
#Set the service provider
INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
#Set the initial context
PROVIDER_URL=ldap://polaris/o=ibm_us,c=us
#Set the authentication type
SECURITY_AUTHENTICATION=none
```

(In this example, a hash sign (#) in the first column of the line indicates a comment, or a line that is not used.)

A sample configuration file, which is used as the default configuration file, is supplied with IBM MQ. The sample file is called JMSAdmin.config, and is found in the <MQ_JAVA_INSTALL_PATH>/bin directory. You can either edit this sample file to define the settings needed for your system, or create your own configuration file.

When you start the administration tool, you can specify the configuration file that you want to use by using the -cfg command-line parameter, as described in "Starting the administration tool" on page 488. If you do not specify a configuration file name when you invoke the tool, the tool attempts to load the default configuration file (JMSAdmin.config). It searches for this file first in the current directory, and then in the <MQ_JAVA_INSTALL_PATH>/bin directory, where <MQ_JAVA_INSTALL_PATH> is the path to your IBM MQ classes for JMS installation.

The names of JMS objects that are stored in an LDAP environment must comply with LDAP naming conventions. One of these conventions is that object and context names must include a prefix, such as cn= (common name), or ou= (organizational unit). The administration tool simplifies the use of LDAP service providers by allowing you to refer to object and context names without a prefix. If you do not supply a prefix, the tool automatically adds a default prefix to the name you supply. For LDAP, this is cn=. If required, you can change the default prefix by setting the **NAME_PREFIX** property in the configuration file.

Note: You might need to configure your LDAP server to store Java objects. For more information, see the documentation for your LDAP server.

Procedure

1. Define the service provider that the tool uses by configuring the **INITIAL_CONTEXT_FACTORY** property.

   The supported values for this property are as follows:

   - com.sun.jndi.ldap.LdapCtxFactory (for LDAP)
   - com.sun.jndi.fscontext.RefFSContextFactory (for file system context)
   - z/OS com.ibm.jndi.LDAPCtxFactory is supported on z/OS only, and provides access to an LDAP server. However, this class is incompatible with com.sun.jndi.ldap.LdapCtxFactory, in that objects created using one InitialContextFactory cannot be read or modified using the other.

   You can also use the administration tool to connect to other JNDI contexts by using three parameters defined in the JMSAdmin configuration file. To use a different InitialContextFactory:

   a) Set the **INITIAL_CONTEXT_FACTORY** property to the required class name.

   b) Define the behavior of the InitialContextFactory using the **USE_INITIAL_DIR_CONTEXT**, **NAME_PREFIX** and **NAME_READABILITY_MARKER** properties.

      The settings for these properties are described in the sample configuration file comments.

   You do not need to define the **USE_INITIAL_DIR_CONTEXT**, **NAME_PREFIX** and **NAME_READABILITY_MARKER** properties if you use one of the supported **INITIAL_CONTEXT_FACTORY** values. However, you can give values to these properties if you want to override the system defaults. For example, if your objects are stored in an LDAP environment, you can change the default prefix that the tool adds to object and context names by setting the **NAME_PREFIX** property to the required prefix.

   If you omit one or more of the three InitialContextFactory properties, the administration tool provides suitable defaults based on the values of the other properties.

2. Define the URL of the initial context of the session by configuring the **PROVIDER_URL** property.

   This URL is the root of all JNDI operations carried out by the tool. Two forms of this property are supported:

   - ldap://hostname/contextname

- file:[drive:]/pathname

The format of the LDAP URL can vary, depending on your LDAP provider. See your LDAP documentation for more information.

3. Define whether JNDI passes security credentials to your service provider by configuring the **SECURITY_AUTHENTICATION** property.

   This property is used only when an LDAP service provider is used and can take one of three values:

   none (anonymous authentication)
      If you set this parameter to `none`, JNDI does not pass any security credentials to the service provider, and *anonymous authentication* is performed.

   `simple` (simple authentication)
      If you set the parameter to `simple`, security credentials are passed through JNDI to the underlying service provider. These security credentials are in the form of a user distinguished name (User DN) and password.

   `CRAM-MD5` (CRAM-MD5 authentication mechanism)
      If you set the parameter to `CRAM-MD5`, security credentials are passed through JNDI to the underlying service provider. These security credentials are in the form of a user distinguished name (User DN) and password.

   If you do not supply a valid value for the **SECURITY_AUTHENTICATION** property, the property defaults to none.

   If security credentials are required, you are prompted for them when the tool initializes. You can avoid this by setting the **PROVIDER_USERDN** and **PROVIDER_PASSWORD** properties in the JMSAdmin configuration file.

   Note: If you do not use these properties, the text typed, *including the password*, is echoed to the screen. This might have security implications.

   The tool does no authentication itself; the authentication task is delegated to the LDAP server. The LDAP server administrator must set up and maintain access privileges to different parts of the directory. See your LDAP documentation for more information. If authentication fails, the tool displays an appropriate error message and terminates.

   More detailed information about security and JNDI is in the documentation at Oracle's Java website (Oracle Technology Network for Java Developers).

Starting the administration tool

The administration tool has a command-line interface that you can use either interactively, or to start a batch process.

About this task

The interactive mode provides a command prompt where you can enter administration commands. In the batch mode, the command to start the tool includes the name of a file that contains an administration command script.

Procedure

Interactive mode

- To start the tool in interactive mode, enter the following command:

  ```
  JMSAdmin [-t] [-v] [-cfg config_filename]
  ```

  where:

  -t
     Enables trace (default is trace off)

The trace file is generated in `"%MQ_JAVA_DATA_PATH%"\errors` (Windows) or `/var/mqm/trace` (UNIX). The name of the trace file is of the form:

`mqjms_ PID.trc`

where *PID* is the process ID of the JVM.

-v
Produces verbose output (default is terse output)

-cfg config_filename
Names an alternative configuration file. If this parameter is omitted, the default configuration file, `JMSAdmin.config`, is used. For more information about the configuration file, see "Configuring the JMS administration tool" on page 486.

A command prompt is displayed, which indicates that the tool is ready to accept administration commands. This prompt initially appears as:

```
InitCtx&gt;
```

indicating that the current context (that is, the JNDI context to which all naming and directory operations currently refer) is the initial context defined in the **PROVIDER_URL** configuration parameter. For more information about this parameter, see "Configuring the JMS administration tool" on page 486.

As you traverse the directory namespace, the prompt changes to reflect this, so that the prompt always displays the current context.

Batch mode

- To start the tool in batch mode, enter the following command:

```
JMSAdmin <test.scp
```

where `test.scp` is a script file that contains administration commands. For more information, see "Using administration commands" on page 489. The last command in the file must be the END command.

Using administration commands

The administration tool accepts commands consisting of an administration verb and its appropriate parameters.

About this task

The following table lists the administration verbs that you can use when entering commands with the administration tool.

Table 32. Administration verbs

Verb	Short form	Description
ALTER	ALT	Change at least one of the properties of an administered object
DEFINE	DEF	Create and store an administered object, or create a subcontext
DISPLAY	DIS	Display the properties of one or more stored administered objects, or the contents of the current context
DELETE	DEL	Remove one or more administered objects from the namespace, or remove an empty subcontext

Table 32. Administration verbs (continued)		
Verb	**Short form**	**Description**
CHANGE	CHG	Alter the current context, allowing the user to traverse the directory namespace anywhere below the initial context (pending security clearance)
COPY	CP	Make a copy of a stored administered object, storing it under an alternative name
MOVE	MV	Alter the name under which an administered object is stored
END		Close the administration tool

Procedure

- If the administration tool is not already started, start it as described in "Starting the administration tool" on page 488.

  The command prompt is displayed, indicating that the tool is ready to accept administration commands. This prompt initially appears as:

  ```
  InitCtx>
  ```

  To change the current context, use the CHANGE verb as described in "Configuring subcontexts" on page 490.
- Enter commands in the following form:

  ```
  verb [param]*
  ```

  where **verb** is one of the administration verbs listed in Table 32 on page 489. All valid commands contain one verb, which appears at the beginning of the command in either its standard or short form. Verb names are not case-sensitive.
- To terminate a command, press Enter, unless you want to enter several commands together, in which case type the plus sign (+) directly before pressing Enter.

  Typically, to terminate commands, you press Enter. However, you can override this by typing the plus sign (+) directly before pressing Enter. This enables you to enter multiline commands, as shown in the following example:

  ```
  DEFINE Q(BookingsInputQueue) +
  QMGR(QM.POLARIS.TEST) +
  QUEUE(BOOKINGS.INPUT.QUEUE) +
  PORT(1415) +
  CCSID(437)
  ```
- To close the administration tool, use the **END** verb.

  This verb cannot take any parameters.

Configuring subcontexts

You can use the verbs **CHANGE**, **DEFINE**, **DISPLAY** and **DELETE** to configure directory namespace subcontexts.

About this task

The use of these verbs is described in the following table.

Table 33. Syntax and description of commands used to manipulate subcontexts	
Command syntax	**Description**
DEFINE CTX(ctxName)	Attempts to create a child subcontext of the current context, having the name ctxName. Fails if there is a security violation, if the subcontext already exists, or if the name supplied is not valid.
DISPLAY CTX	Displays the contents of the current context. Administered objects are annotated with a, subcontexts with `[D]`. The Java type of each object is also displayed.
DELETE CTX(ctxName)	Attempts to delete the current context's child context having the name ctxName. Fails if the context is not found, is non-empty, or if there is a security violation.
CHANGE CTX(ctxName)	Alters the current context, so that it now refers to the child context having the name ctxName. One of two special values of ctxName can be supplied: **=UP** moves to the parent of the current context **=INIT** moves directly to the initial context Fails if the specified context does not exist, or if there is a security violation.

The names of JMS objects that are stored in an LDAP environment must comply with LDAP naming conventions. One of these conventions is that object and context names must include a prefix, such as `cn=` (common name), or ou= (organizational unit). The administration tool simplifies the use of LDAP service providers by allowing you to refer to object and context names without a prefix. If you do not supply a prefix, the tool automatically adds a default prefix to the name you supply. For LDAP, this is `cn=`. If required, you can change the default prefix by setting the **NAME_PREFIX** property in the configuration file. For more information, see "Configuring the JMS administration tool" on page 486.

Note: You might need to configure your LDAP server to store Java objects. For more information, see the documentation for your LDAP server.

Creating JMS objects

To create JMS connection factory and destination objects and store them in a JNDI namespace, use the `DEFINE` verb. To store your objects in an LDAP environment, you must give them names that comply with certain conventions. The administration tool can help you obey LDAP naming conventions by adding a default prefix to object names.

About this task

The DEFINE verb creates an administered object with the type, name and properties that you specify. The new object is stored in the current context.

The names of JMS objects that are stored in an LDAP environment must comply with LDAP naming conventions. One of these conventions is that object and context names must include a prefix, such as `cn=` (common name), or ou= (organizational unit). The administration tool simplifies the use of LDAP service providers by allowing you to refer to object and context names without a prefix. If you do not supply a prefix, the tool automatically adds a default prefix to the name you supply. For LDAP, this is `cn=`. If required, you can change the default prefix by setting the **NAME_PREFIX** property in the configuration file. For more information, see "Configuring the JMS administration tool" on page 486.

Note: You might need to configure your LDAP server to store Java objects. For more information, see the documentation for your LDAP server.

Procedure

1. If the administration tool is not already started, start it as described in “Starting the administration tool” on page 488.

   The command prompt is displayed, indicating that the tool is ready to accept administration commands.
2. Make sure that command prompt is showing the context in which you want to create the new object.

   When you start the administration tool, the prompt initially appears as:

   ```
   InitCtx>
   ```

   To change the current context, use the CHANGE verb as described in “Configuring subcontexts” on page 490.
3. To create a connection factory, queue destination or topic destination, use the following command syntax:

   ```
   DEFINE TYPE (name) [property]*
   ```

   That is, type the DEFINE verb, followed by a *TYPE* (name) administered object reference, followed by zero or more *properties* (see Properties of IBM MQ classes for JMS objects).
4. To create a connection factory, queue destination or topic destination, use the following command syntax:

   ```
   DEFINE TYPE (name) [property]*
   ```
5. To display the newly created object, use the DISPLAY verb with the following command syntax:

   ```
   DISPLAY TYPE (name)
   ```

Example

The following example shows a queue called testQueue created in the initial context using the DEFINE verb. Since this object is being stored in an LDAP environment, although the object name testQueue is not entered with a prefix, the tool automatically adds one to ensure compliance with the LDAP naming convention. Submitting the command DISPLAY Q(testQueue) also causes this prefix to be added.

```
InitCtx> DEFINE Q(testQueue)

InitCtx> DISPLAY CTX

Contents of InitCtx

a cn=testQueue       com.ibm.mq.jms.MQQueue

1 Object(s)
0 Context(s)
1 Binding(s), 1 Administered
```

Sample error conditions creating a JMS object

A number of common error conditions can arise when you create an object.

The following are examples of these error conditions:

CipherSpec mapped to CipherSuite

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) SSLCIPHERSUITE(RC4_MD5_US)
WARNING: Converting CipherSpec RC4_MD5_US to
CipherSuite SSL_RSA_WITH_RC4_128_MD5
```

Invalid property for object

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) PRIORITY(4)
Unable to create a valid object, please check the parameters supplied
Invalid property for a QCF: PRI
```

Invalid type for property value

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) CCSID(english)
Unable to create a valid object, please check the parameters supplied
Invalid value for CCS property: English
```

Property clash - client/bindings

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) HOSTNAME(polaris.hursley.ibm.com)
Unable to create a valid object, please check the parameters supplied
Invalid property in this context: Client-bindings attribute clash
```

Property clash - Exit initialization

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) SECEXITINIT(initStr)
Unable to create a valid object, please check the parameters supplied
Invalid property in this context: ExitInit string supplied
without Exit string
```

Property value outside valid range

```
InitCtx/cn=Trash> DEFINE Q(testQ) PRIORITY(12)
Unable to create a valid object, please check the parameters supplied
Invalid value for PRI property: 12
```

Unknown property

```
InitCtx/cn=Trash> DEFINE QCF(testQCF) PIZZA(ham and mushroom)
Unable to create a valid object, please check the parameters supplied
Unknown property: PIZZA
```

The following are examples of error conditions that might arise on Windows when looking up JNDI administered objects from a JMS application.

1. If you are using the WebSphere JNDI provider, com.ibm.websphere.naming.WsnInitialContextFactory, you must use a forward slash (/) to access administered objects defined in subcontexts; for example, jms/MyQueueName. If you use a backslash (\), an InvalidNameException is thrown.
2. If you are using the Oracle JNDI provider, com.sun.jndi.fscontext.RefFSContextFactory, you must use a backslash (\) to access administered objects defined in subcontexts; for example, ctx1\\fred. If you use a forward slash (/), a NameNotFoundException is thrown.

Configuring JMS objects

You can use the verbs `ALTER`, `DEFINE`, `DISPLAY`, `DELETE`, `COPY`, and `MOVE` to manipulate administered objects in the directory namespace.

About this task

Table 34 on page 494 summarizes the use of these verbs. Substitute *TYPE* with the keyword that represents the required administered object, as described in "Configuring JMS objects using the administration tool" on page 485.

Table 34. Syntax and description of commands used to manipulate administered objects	
Command syntax	**Description**
ALTER *TYPE* (name) [property]*	Attempts to update the properties of the administered object with the ones supplied. Fails if there is a security violation, if the specified object cannot be found, or if the new properties supplied are not valid.
DEFINE *TYPE* (name) [property]*	Attempts to create an administered object of type *TYPE* with the supplied properties, and store it under the name `name` in the current context. Fails if there is a security violation, if the supplied name is not valid or an object of that name exists, or if the properties supplied are not valid.
DISPLAY *TYPE* (name)	Displays the properties of the administered object of type *TYPE* , bound under the name `name` in the current context. Fails if the object does not exist, or if there is a security violation.
DELETE *TYPE* (name)	Attempts to remove the administered object of type *TYPE* , having the name `name`, from the current context. Fails if the object does not exist, or if there is a security violation.
COPY *TYPE* (nameA) *TYPE* (nameB)	Makes a copy of the administered object of type *TYPE* , having the name `nameA`, naming the copy `nameB`. This all occurs within the scope of the current context. Fails if the object to be copied does not exist, if an object of name nameB exists, or if there is a security violation.
MOVE *TYPE* (nameA) *TYPE* (nameB)	Moves (renames) the administered object of type *TYPE* , having the name `nameA`, to `nameB`. This all occurs within the scope of the current context. Fails if the object to be moved does not exist, if an object of name nameB exists, or if there is a security violation.

Configuring JMS resources in WebSphere Application Server

To configure JMS resources in WebSphere Application Server, you can either use the administrative console or wsadmin commands.

About this task

Java Message Service (JMS) applications typically rely on externally configured objects which describe how the application connects to its JMS provider and the destinations it accesses. JMS applications use the Java Naming and Directory Interface (JNDI) to access the following types of object at runtime:

- Activation specifications (used by Java EE application servers)
- Unified connection factories (with JMS 1.1, domain-independent (unified) connection factories are preferred to domain-specific queue connection factories and topic connection factories)
- Topic connection factories (used by JMS 1.0 applications)
- Queue connection factories (used by JMS 1.0 applications)
- Queues
- Topics

Through the IBM MQ messaging provider in WebSphere Application Server, Java Message Service (JMS) messaging applications can use your IBM MQ system as an external provider of JMS messaging resources. To enable this approach, you configure the IBM MQ messaging provider in WebSphere Application Server to define JMS resources for connecting to any queue manager on the IBM MQ network.

You can use WebSphere Application Server to configure IBM MQ resources for applications (for example queue connection factories) and to manage messages and subscriptions associated with JMS destinations. You administer security through IBM MQ.

Related information for WebSphere Application Server Version 8.5.5
Interoperation using the IBM MQ messaging provider
Managing messaging with the IBM MQ messaging provider
Mapping of administrative console panel names to command names and IBM MQ names
Related information for WebSphere Application Server Version 8.0
Interoperation using the IBM MQ messaging provider
Managing messaging with the IBM MQ messaging provider
Mapping of administrative console panel names to command names and IBM MQ names
Related information for WebSphere Application Server Version 7.0
Interoperation using the IBM MQ messaging provider
Managing messaging with the IBM MQ messaging provider
Mapping of administrative console panel names to command names and IBM MQ names

Configuring JMS resources using the administrative console

You can use the WebSphere Application Server administrative console to configure activation specifications, connection factories and destinations for the IBM MQ JMS provider.

About this task

You can use the WebSphere Application Server administrative console to create, view, or modify any of the following resources:

- Activation specifications
- Domain-independent connection factories (JMS 1.1 or later)
- Queue connection factories
- Topic connection factories
- Queues
- Topics

The following steps provide an overview of the ways in which you can use the administrative console to configure JMS resources for use with the IBM MQ messaging provider. Each step includes the name of the topic in the WebSphere Application Server product documentation to which you can refer for more information. See *Related links* for links to these topics in the WebSphere Application Server 8.5.5, Version 8.0 and Version 7.0 product documentation.

In a mixed-version WebSphere Application Server cell, you can administer IBM MQ resources on nodes of all versions. However, some properties are not available on all versions. In this situation, only the properties of that particular node are displayed in the administrative console.

Procedure

To create or configure an activation specification for use with the IBM MQ messaging provider:

- To create an activation specification, use the Create IBM MQ JMS resource wizard.

  You can either use the wizard to specify all the details for the activation specification, or you can choose to specify the connection details for the IBM MQ by using a client channel definition table (CCDT). When you specify the connection details using the wizard, you can choose either to enter host and port information separately or, if you are using a multi-instance queue manager, to enter host and port information in the form of a connection name list. For more information, see *Creating an activation specification for the IBM MQ messaging provider*.

- To view or change the configuration properties of an activation specification, use the administrative console IBM MQ messaging provider connection factory settings panel.

  These configuration properties control how connections are created to associated queues and topics. For more information, see *Configuring an activation specification for the IBM MQ messaging provider*.

To create or configure a unified connection factory, a queue connection factory, or a topic connection factory for use with the IBM MQ messaging provider:

- To create a connection factory, first select the type of connection factory that you want to create, then use the Create IBM MQ JMS resource wizard to specify the details.
  - If your JMS application is intended to use only point-to-point messaging, create a domain-specific connection factory for the point-to-point messaging domain that can be used for creating connections specifically for point-to-point messaging.
  - If your JMS application is intended only to use publish/subscribe messaging, create a domain-specific connection factory for the publish/subscribe messaging domain that can be used for creating connections specifically for publish/subscribe messaging.
  - For JMS 1.1 or later, create a domain-independent connection factory that can be used for both point-to-point messaging and publish/subscribe messaging, allowing your application to perform both point-to-point and publish/subscribe work under the same transaction.

  You can choose whether to use the wizard to specify all the details for the connection factory, or you can choose to specify the connection details for the IBM MQ by using a client channel definition table (CCDT). When you specify the connection details using the wizard, you can choose either to enter host and port information separately or, if you are using a multi-instance queue manager, to enter host and port information in the form of a connection name list. For more information, see *Creating a connection factory for the IBM MQ messaging provider*.

To view or change the configuration properties of a connection factory:

- Use the administrative console connection factory settings panel for the type of connection factory that you want to configure.

  The configuration properties control how connections are created to associated queues and topics. For more information, see *Configuring a collection factory for the IBM MQ messaging provider*, or *Configuring a queue collection factory for the IBM MQ messaging provider*, or *Configuring a topic collection factory for the IBM MQ messaging provider*.

To configure a JMS queue destination for point-to-point messaging with the IBM MQ messaging provider:

- Use the administrative console IBM MQ messaging provider queue settings panel to define the following types of property:
  - General properties, including administration and IBM MQ queue properties.
  - Connection properties that specify how to connect to the queue manager that hosts the queue.
  - Advanced properties that control the behavior of connections made to IBM MQ messaging provider destinations.
  - Any custom properties for the queue destination.

  For more information, see *Configuring a queue for the IBM MQ messaging provider*.

To create or configure a JMS topic destination for publish/subscribe messaging with the IBM MQ messaging provider:

- Use the IBM MQ messaging provider topic settings panel to define the following types of property:
  - General properties, including administration and IBM MQ topic properties.
  - Advanced properties that control the behavior of connections made to IBM MQ messaging provider destinations.
  - Any custom properties for the queue destination.

  For more information, see *Configuring a topic for the IBM MQ messaging provider*.

Related concepts

"Client channel definition table" on page 38

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On platforms other than z/OS a CCDT is created automatically. You must then make it available to the client application.

"Multi-instance queue managers" on page 402
Multi-instance queue managers are instances of the same queue manager configured on different servers. One instance of the queue manager is defined as the active instance and another instance is defined as the standby instance. If the active instance fails, the multi-instance queue manager restarts automatically on the standby server.

"Configuring publish/subscribe messaging" on page 331
You can start, stop and display the status of queued publish/subscribe. You can also add and remove streams, and add and delete queue managers from a broker hierarchy.

Related information for WebSphere Application Server Version 8.5.5

IBM MQ messaging provider activation specifications
Creating an activation specification for the IBM MQ messaging provider
Configuring an activation specification for the IBM MQ messaging provider
Creating a connection factory for the IBM MQ messaging provider
Configuring a unified connection factory for the IBM MQ messaging provider
Configuring a queue connection factory for the IBM MQ messaging provider
Configuring a topic connection factory for the IBM MQ messaging provider
Configuring a queue for the IBM MQ messaging provider
Configuring a topic for the IBM MQ messaging provider

Related information for WebSphere Application Server Version 8.0

IBM MQ messaging provider activation specifications
Creating an activation specification for the IBM MQ messaging provider
Configuring an activation specification for the IBM MQ messaging provider
Creating a connection factory for the IBM MQ messaging provider
Configuring a unified connection factory for the IBM MQ messaging provider
Configuring a queue connection factory for the IBM MQ messaging provider
Configuring a topic connection factory for the IBM MQ messaging provider
Configuring a queue for the IBM MQ messaging provider
Configuring a topic for the IBM MQ messaging provider

Related information for WebSphere Application Server Version 7.0

IBM MQ messaging provider activation specifications
Creating an activation specification for the IBM MQ messaging provider
Configuring an activation specification for the IBM MQ messaging provider
Creating a connection factory for the IBM MQ messaging provider
Configuring a unified connection factory for the IBM MQ messaging provider
Configuring a queue connection factory for the IBM MQ messaging provider
Configuring a topic connection factory for the IBM MQ messaging provider
Configuring a queue for the IBM MQ messaging provider
Configuring a topic for the IBM MQ messaging provider

Configuring JMS resources using wsadmin scripting commands

You can use WebSphere Application Server wsadmin scripting commands to create, modify, delete or show information about JMS activation specifications, connection factories, queues and topics. You can also display and manage the settings for the IBM MQ resource adapter.

About this task

The following steps provide an overview of the ways in which you can use WebSphere Application Server wsadmin commands to configure JMS resources for use with the IBM MQ messaging provider. For more information about how to use these commands, see *Related links* for links to the WebSphere Application Server Version 8.5.5, Version 8.0 and Version 7.0 product documentation.

To run a command, use the AdminTask object of the wsadmin scripting client.

After using a command to create a new object or make changes, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

To see a list of the available IBM MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

To see overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Procedure

To list all of the IBM MQ messaging provider resources defined at the scope at which a command is issued, use the following commands.

- To list the activation specifications, use the **`listWMQActivationSpecs`** command.
- To list the connection factories, use the **`listWMQConnectionFactories`** command.
- To list the queue type destinations, use the **`listWMQQueues`** command.
- To list the topic type destinations, use the **`listWMQTopics`** command.

To create a JMS resource for the IBM MQ messaging provider at a specific scope, use the following commands.

- To create an activation specification, use the **`createWMQActivationSpec`** command.

  You can either create an activation specification by specifying all the parameters to be used for establishing a connection, or you can create the activation specification so that it uses a client channel definition table (CCDT) to locate the queue manager to connect to.
- To create a connection factory, use the **`createWMQConnectionFactory`** command, using the **`-type`** parameter to specify the type of connection factory that you want to create:
  - If your JMS application is intended to use only point-to-point messaging, create a domain-specific connection factory for the point-to-point messaging domain that can be used for creating connections specifically for point-to-point messaging.
  - If your JMS application is intended only to use publish/subscribe messaging, create a domain-specific connection factory for the publish/subscribe messaging domain that can be used for creating connections specifically for publish/subscribe messaging.
  - For JMS 1.1 or later, create a domain-independent connection factory that can be used for both point-to-point messaging and publish/subscribe messaging, allowing your application to perform both point-to-point and publish/subscribe work under the same transaction.

  The default type is domain-independent connection factory.
- To create a queue type destination, use the **`createWMQQueue`** command.
- To create a topic type destination, use the **`createWMQTopic`** command.

To modify a JMS resource for the IBM MQ messaging provider at a specific scope, use the following commands.

- To modify an activation specification, use the **`modifyWMQActivationSpec`** command.

  You cannot change the type of an activation specification. For example, you cannot create an activation specification where you enter all the configuration information manually and then modify it to use a CCDT.

- To modify a connection factory, use the **modifyWMQConnectionFactory** command.
- To modify a queue type destination, use the **modifyWMQQueue** command.
- To modify a topic type destination, use the **modifyWMQTopic** command.

To delete a JMS resource for the IBM MQ messaging provider at a specific scope, use the following commands.

- To delete an activation specification, use the **deleteWMQActivationSpec** command.
- To delete a connection factory, use the **deleteWMQConnectionFactory** command.
- To delete a queue type destination, use the **deleteWMQQueue** command.
- To delete a topic type destination, use the **deleteWMQTopic** command.

To display information about a specific IBM MQ messaging provider resource, use the following commands.

- To display all the parameters, and their values, associated with a particular activation specification, use the **showWMQActivationSpec** command.
- To display all the parameters, and their values, associated with a particular connection factory, use the **showWMQConnectionFactory** command.
- To display all the parameters, and their values, associated with a particular queue type destination, use the **showWMQQueue** command.
- To display all the parameters, and their values, associated with a topic type destination, use the **deleteWMQTopic** command.

To manage settings for the IBM MQ resource adapter or the IBM MQ messaging provider, use the following commands.

- To manage the settings of the IBM MQ resource adapter that is installed at a particular scope, use the **manageWMQ** command.
- To display all the parameters, and their values that can be set by the **manageWMQ** command, use the **showWMQ** command. These settings are either related to the IBM MQ resource adapter or the IBM MQ messaging provider. The **showWMQ** command also shows any custom properties that are set on the IBM MQ resource adapter.

Related concepts

"Client channel definition table" on page 38
The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On platforms other than z/OS a CCDT is created automatically. You must then make it available to the client application.

"Multi-instance queue managers" on page 402
Multi-instance queue managers are instances of the same queue manager configured on different servers. One instance of the queue manager is defined as the active instance and another instance is defined as the standby instance. If the active instance fails, the multi-instance queue manager restarts automatically on the standby server.

"Configuring publish/subscribe messaging" on page 331
You can start, stop and display the status of queued publish/subscribe. You can also add and remove streams, and add and delete queue managers from a broker hierarchy.

Related information for WebSphere Application Server Version 8.5.5

createWMQActivationSpec command
createWMQConnectionFactory command
createWMQQueue command
createWMQTopic command
deleteWMQActivationSpec command
deleteWMQConnectionFactory command
deleteWMQQueue command
deleteWMQTopic command

listWMQActivationSpecs command
listWMQConnectionFactories command
listWMQQueues command
listWMQTopics command
modifyWMQActivationSpec command
modifyWMQConnectionFactory command
modifyWMQQueue command
modifyWMQTopic command
showWMQActivationSpec command
showWMQConnectionFactory command
showWMQQueue command
showWMQTopic command
showWMQ command
manageWMQ command
Related information for WebSphere Application Server Version 8.0
createWMQActivationSpec command
createWMQConnectionFactory command
createWMQQueue command
createWMQTopic command
deleteWMQActivationSpec command
deleteWMQConnectionFactory command
deleteWMQQueue command
deleteWMQTopic command
listWMQActivationSpecs command
listWMQConnectionFactories command
listWMQQueues command
listWMQTopics command
modifyWMQActivationSpec command
modifyWMQConnectionFactory command
modifyWMQQueue command
modifyWMQTopic command
showWMQActivationSpec command
showWMQConnectionFactory command
showWMQQueue command
showWMQTopic command
showWMQ command
manageWMQ command
Related information for WebSphere Application Server Version 7.0
createWMQActivationSpec command
createWMQConnectionFactory command
createWMQQueue command
createWMQTopic command
deleteWMQActivationSpec command
deleteWMQConnectionFactory command
deleteWMQQueue command
deleteWMQTopic command
listWMQActivationSpecs command
listWMQConnectionFactories command
listWMQQueues command

listWMQTopics command
modifyWMQActivationSpec command
modifyWMQConnectionFactory command
modifyWMQQueue command
modifyWMQTopic command
showWMQActivationSpec command
showWMQConnectionFactory command
showWMQQueue command
showWMQTopic command
showWMQ command
manageWMQ command

Configuring the application server to use the latest resource adapter maintenance level

To ensure that the IBM MQ resource adapter is automatically updated to the latest available maintenance level when you apply WebSphere Application Server fix packs, you can configure all servers in your environment to use the latest version of the resource adapter contained in the WebSphere Application Server fix pack that you have applied to the installation of each node.

Before you begin

Important: If you are using WebSphere Application Server Version 7.0, Version 8 or Version 8.5 on any platform, do not install the IBM MQ 8.0 resource adapter into the application server. The IBM MQ 8.0 resource adapter can only be deployed into an application server that supports JMS 2.0. However, WebSphere Application Server Version 7.0, Version 8 and Version 8.5 only support JMS 1.1. These versions of WebSphere Application Server come with the IBM WebSphere MQ 7.0 resource adapter, which can be used to connect to a IBM MQ 8.0 queue manager using either the BINDINGS or CLIENT transport.

About this task

Use this task if any of the following circumstances apply to your configuration, and you want to configure all servers in your environment to use the latest version of the IBM MQ resource adapter:

- The JVM logs of any application server in your environment show the following IBM MQ resource adapter version information after WebSphere Application Server Version 7.0 Fix Pack 1 or later has been applied:

  ```
  WMSG1703I:RAR implementation Version 7.0.0.0-k700-L080820
  ```
- The JVM logs of any application server in your environment contain the following entry:

  ```
  WMSG1625E: It was not possible to detect
  the IBM MQ messaging provider code at the specified path <null>
  ```
- One or more nodes has previously been manually updated to use a specific maintenance level of the IBM MQ resource adapter that is now superseded by the latest version of the resource adapter contained in the current WebSphere Application Server maintenance level.

The *profile_root* directory that the examples refer to is the home directory for the WebSphere Application Server profile, for example `C:\Program Files\IBM\WebSphere\AppServer1`.

When you have performed the following steps for all cells and single server installations in your environment, your servers automatically receive maintenance to the IBM MQ resource adapter when a new WebSphere Application Server fix pack is applied.

Procedure

1. Start the application server. If the profile is part of a network deployment configuration, start the deployment manager and all node agents. If the profile contains an administrative agent, start the administrative agent.

2. Check the maintenance level of the IBM MQ resource adapter.
   a) Open a command prompt window and change to the *profile_root*\bin directory.

      For example, enter cd C:\Program Files\IBM\WebSphere\AppServer1\bin.
   b) Start the wsadmin tool by entering wsadmin.bat -lang jython, then if prompted to do so, enter your username and password.
   c) Type the following command, then press Return twice:

```
wmqInfoMBeansUnsplit = AdminControl.queryNames("WebSphere:type=WMQInfo,*")
wmqInfoMBeansSplit = AdminUtilities.convertToList(wmqInfoMBeansUnsplit)
for wmqInfoMBean in wmqInfoMBeansSplit: print wmqInfoMBean; print AdminControl.invoke(wmqInfoMBean,
'getInfo', '')
```

      You can also run this command in Jacl. For further information about how to do this, see *Ensuring that servers use the latest available IBM MQ resource adapter maintenance level* in the WebSphere Application Server product documentation.
   d) Find the WMSG1703I message in the displayed output from the command and check the resource adapter level.

      For example, for WebSphere Application Server Version 7.0 Fixpack 15, the message should be:

      `WMSG1703I: RAR implementation Version 7.0.1.3-k701-103-100812`

      This message shows that the version is 7.0.1.3-k701-103-100812, which is the correct resource adapter level for this fixpack. However, if the following message is displayed instead, this means that you need to adjust the resource adapter to the correct level of maintenance for Fix Pack 15.

      `WMSG1703I: RAR implementation Version 7.0.0.0-k700-L080820`
3. Copy the following Jython script into a file called convertWMQRA.py, then save it into the profile root directory, for example C:\Program Files\IBM\WebSphere\AppServer1\bin.

```
ras = AdminUtilities.convertToList(AdminConfig.list('J2CResourceAdapter'))

for ra in ras :
  desc = AdminConfig.showAttribute(ra, "description")
  if (desc == "WAS 7.0 Built In IBM MQ Resource Adapter") or (desc == "WAS 7.0.0.1 Built In IBM MQ
Resource Adapter"):
    print "Updating archivePath and classpath of " + ra
    AdminConfig.modify(ra, [['archivePath', "${WAS_INSTALL_ROOT}/installedConnectors/wmq.jmsra.rar"]])
    AdminConfig.unsetAttributes(ra, ['classpath'])
    AdminConfig.modify(ra, [['classpath', "${WAS_INSTALL_ROOT}/installedConnectors/wmq.jmsra.rar"]])
    AdminConfig.save()
  #end if
#end for
```

   Tip: When saving the file, make sure that it is saved as a python file rather than a text file.
4. Use the WebSphere Application Server wsadmin tool to run the Jython script that you have just created.

   Open a command prompt and navigate to the \bin directory in the home directory for the WebSphere Application Server, for example C:\Program Files\IBM\WebSphere\AppServer1\bin directory, then type the following command and press Return:

```
wsadmin -lang jython -f convertWMQRA.py
```

   If prompted to do so, enter your username and password.

   Note: If you run the script against a profile that is part of a network deployment configuration, the script updates all profiles that need updating in that configuration. A full resynchronization might be necessary if you have pre-existing configuration file inconsistencies.
5. If you are running in a network deployment configuration, ensure that the node agents are fully re-synchronized. For more information, see Synchronizing nodes using the wsadmin scripting tool or Adding, managing, and removing nodes.
6. Stop all servers in the profile. If the profile is part of a network deployment configuration, also stop any cluster members in the configuration, stop all node agents in the configuration, and stop the deployment manager. If the profile contains an administrative agent, stop the administrative agent.

7. Run the **osgiCfgInit** command from the *profile_root*/bin directory.

   The osgiCfgInit command resets the class cache used by the OSGi runtime environment. If the profile is part of a network deployment configuration, run the **osgiCfgInit** command from the profile_root/bin directory of every profile that is part of the configuration.
8. Restart all servers in the profile. If the profile is part of a network deployment configuration, also restart any cluster members in the configuration, restart all node agents in the configuration, and restart the deployment manager. If the profile contains an administrative agent, restart the administrative agent.
9. Repeat step 2 to check that the resource adapter is now at the correct level.

What to do next

If you continue to experience problems after performing the steps described in this topic, and you have previously used the **Update resource adapter** button on the JMS Provider Settings panel in the WebSphere Application Server administrative console to update the IBM MQ resource adapter on any nodes in your environment, it is possible that you are experiencing the issue described in APAR PM10308.

Related tasks

Using the IBM MQ resource adapter

Related information for WebSphere Application Server Version 8.5.5

Ensuring that servers use the latest available IBM MQ resource adapter maintenance level

Synchronizing nodes using the wsadmin scripting tool

Adding, managing, and removing nodes

JMS provider settings

Related information for WebSphere Application Server Version 8.0

Ensuring that servers use the latest available IBM MQ resource adapter maintenance level

Synchronizing nodes using the wsadmin scripting tool

Adding, managing, and removing nodes

JMS provider settings

Related information for WebSphere Application Server Version 7.0

Ensuring that servers use the latest available IBM MQ resource adapter maintenance level

Synchronizing nodes using the wsadmin scripting tool

Adding, managing, and removing nodes

JMS provider settings

Configuring the JMS PROVIDERVERSION property

The IBM MQ messaging provider has three modes of operation: normal mode, normal mode with restrictions, and migration mode. You can set the JMS **PROVIDERVERSION** property to select which of these modes a JMS application uses to publish and subscribe.

About this task

The selection of the IBM MQ messaging provider mode of operation can be primarily controlled by setting the PROVIDERVERSION connection factory property. The mode of operation can also be selected automatically if a mode has not been specified.

The **PROVIDERVERSION** property differentiates between the three IBM MQ messaging provider modes of operation:

IBM MQ messaging provider normal mode
: Normal mode uses all the features of an IBM MQ queue manager to implement JMS. This mode is optimized to use the JMS 2.0 API and functionality.

IBM MQ messaging provider normal mode with restrictions
: Normal mode with restrictions uses the JMS 2.0 API, but not the new features, that is, shared subscriptions, delayed delivery, and asynchronous send.

IBM MQ messaging provider migration mode

With migration mode, you can connect to a IBM MQ 8.0 queue manager, but none of the features of a IBM WebSphere MQ 7.0 or later queue manager, such as read ahead and streaming, are used.

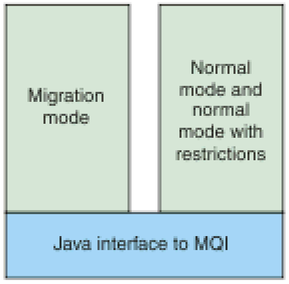

Figure 84. Messaging provider modes

Procedure

To configure the **PROVIDERVERSION** property for a specific connection factory:

- To configure the **PROVIDERVERSION** property using MQ Explorer, see Configuring queue managers and objects.
- To configure the **PROVIDERVERSION** property using the JMS administration tool, see Configuring queue managers and objects.
- To configure the **PROVIDERVERSION** property in a JMS application using the IBM JMS extensions or IBM MQ JMS extensions, seeCreating and configuring connection factories and destinations in an IBM MQ classes for JMS application.

To override connection factory provider mode settings for all connection factories in the JVM:

- To override connection factory provider mode settings, use the `com.ibm.msg.client.wmq.overrideProviderVersion` property

  If you cannot change the connection factory that you are using, you can use the `com.ibm.msg.client.wmq.overrideProviderVersion` property to override any setting on the connection factory. This override applies to all connection factories in the JVM but the actual connection factory objects are not modified.

Related tasks

JMS provider version troubleshooting

Related reference

PROVIDERVERSION

Connection factory properties

Dependencies between properties of IBM MQ classes for JMS objects

IBM MQ messaging provider modes of operation

You can select which of these modes of operation a JMS application uses to publish and subscribe by setting the PROVIDERVERSION property for the connection factory to the appropriate value. In some cases, the PROVIDERVERSION property is set as unspecified, in which case the JMS client uses an algorithm to determine which mode of operation to use.

PROVIDERVERSION property values

You can set the connection factory **PROVIDERVERSION** property to any of the following values:

8 - normal mode

The JMSApplication uses normal mode. This mode uses all the features of an IBM MQ queue manager to implement JMS.

7 - normal mode with restrictions
The JMSapplication uses normal mode with restrictions. This mode uses the JMS 2.0 API, but not the new features such as shared subscriptions, delayed delivery, or asynchronous send.

6- migration mode
The JMS application uses migration mode. In migration mode, the IBM MQ classes for JMS use the features and algorithms similar to those that are supplied with IBM WebSphere MQ 6.0.

`unspecified` (the default value)
The JMS client uses an algorithm to determine which mode of operation is used.

The value that you specify for the **PROVIDERVERSION** property must be a string. If you are specifying an option of 8, 7 or 6, you can do this in any of the following formats:

- V.R.M.F
- V.R.M
- V.R
- V

where V, R, M and F are integer values greater than or equal to zero. The extra R, M and F values are optional and are available for you to use in case fine grained control is needed. For example, if you wanted to use a **PROVIDERVERSION** level of 7, you could set **PROVIDERVERSION** = `7`, `7.0`, `7.0.0` or `7.0.0.0`.

Types of connection factory object

You can set the **PROVIDERVERSION** property for the following types of connection factory object:

- MQConnectionFactory
- MQQueueConnectionFactory
- MQTopicConnectionFactory
- MQXAConnectionFactory
- MQXAQueueConnectionFactory
- MQXAQueueConnectionFactory
- MQXAQueueConnectionFactory
- MQXATopicConnectionFactory

For more information about these different types of connection factory, see "Configuring JMS objects using the administration tool" on page 485.

Related concepts
IBM MQ classes for JMS architecture

PROVIDERVERSION normal mode
Normal mode uses all the features of an IBM MQ queue manager to implement JMS. This mode is optimized to use the JMS 2.0 API and functionality.

The following flowchart shows the checks that the JMS client makes to determine whether a normal mode connection can be created.

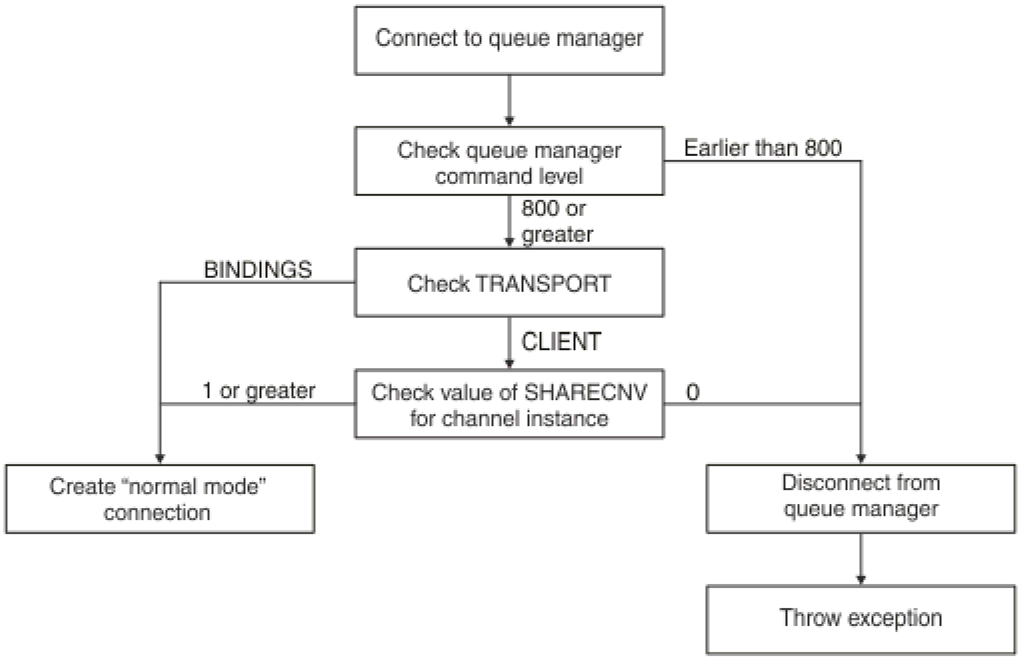

Figure 85. PROVIDERVERSION normal mode

If the queue manager specified in the connection factory settings has a command level of 800 or greater, and the **TRANSPORT** property of the connection factory is set to BINDINGS, a normal mode connection is created without checking any further properties.

If the queue manager specified in the connection factory settings has a command level of 800 or greater, and the **TRANSPORT** property is set to CLIENT, the **SHARECNV** property on the server connection channel is also checked. This check is needed because IBM MQ messaging provider normal mode uses the sharing conversations feature. Therefore, for a normal mode connection attempt to be successful, the **SHARECNV** property, which controls the number of conversations that can be shared, must have a value of 1 or greater.

If all the checks shown in the flowchart are successful, a normal mode connection to the queue manager is created and all of the JMS 2.0 API and features, that is, asynchronous send, delayed delivery, and shared subscription, can then be used.

An attempt to create a normal mode connection fails for either of the following reasons:

- The queue manager specified in the connection factory settings has a command level that is earlier than 800. In this case, the `createConnection` method fails with an exception `JMSFMQ0003`.
- The **SHARECNV** property on the server connection channel is set to `0`. If this property does not have a value of 1 or greater, the `createConnection` method fails with an exception `JMSCC5007`.

Related reference

Dependencies between properties of IBM MQ classes for JMS objects

DEFINE CHANNEL (SHARECNV property)

TRANSPORT

PROVIDERVERSION normal mode with restrictions

Normal mode with restrictions uses the JMS 2.0 API, but not the new IBM MQ 8.0 features such as shared subscriptions, delayed delivery, or asynchronous send.

The following flowchart shows the checks that the JMS client makes to determine whether a normal mode with restrictions connection can be created .

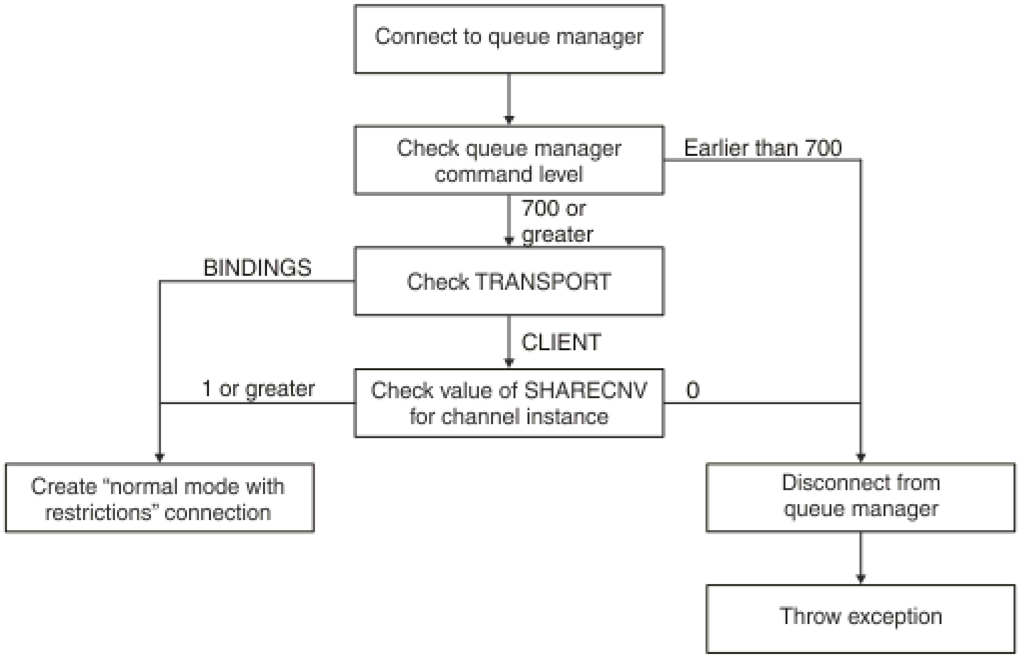

Figure 86. PROVIDERVERSION normal mode with restrictions

If the queue manager specified in the connection factory settings has a command level of 700 or greater, and the **TRANSPORT** property of the connection factory is set to BINDINGS, a normal mode connection is created without checking any further properties.

If the queue manager specified in the connection factory settings has a command level of 700 or greater, and the **TRANSPORT** property is set to CLIENT, the **SHARECNV** property on the server connection channel is also checked. This check is needed because IBM MQ messaging provider normal mode with restrictions uses the sharing conversations feature. Therefore, for a normal mode with restrictions connection attempt to be successful, the **SHARECNV** property, which controls the number of conversations that can be shared, must have a value of 1 or greater.

If all the checks shown in the flowchart are successful, a normal mode with restrictions connection to the queue manager is created and you can then use the JMS 2.0 API, but not the asynchronous send, delayed delivery, or shared subscription features.

An attempt to create a normal mode with restrictions connection fails for either of the following reasons:

- The queue manager specified in the connection factory settings has a command level that is earlier than 700. In this case, the `createConnection` method fails with exception JMSFCC5008.
- The **SHARECNV** property on the server connection channel is set to `0`. If this property does not have a value of 1 or greater, the `createConnection` method fails with an exception `JMSCC5007`.

Related reference

Dependencies between properties of IBM MQ classes for JMS objects
DEFINE CHANNEL (SHARECNV property)
TRANSPORT

PROVIDERVERSION migration mode

For migration mode, the IBM MQ classes for JMS use the features and algorithms similar to those that are supplied with IBM WebSphere MQ 6.0, such as queued publish/subscribe, selection implemented on the client side, non-multiplex channels, and polling used to implement listeners.

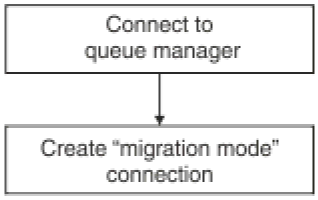

Figure 87. PROVIDERVERSION migration mode

If you want to connect to WebSphere Message Broker Version 6.0 or 6.1 using IBM MQ Enterprise Transport Version 6.0, you must use migration mode.

You can connect to an IBM MQ 8.0 queue manager using migration mode, but none of the new features of an IBM MQ classes for JMS queue manager are used, for example, read ahead or streaming. If you have an IBM MQ 8.0 client connecting to an IBM MQ 8.0 queue manager on a distributed platform or an IBM MQ 8.0 queue manager on z/OS, then the message selection is done by the queue manager rather than on the client system.

If IBM MQ messaging provider migration mode is specified and the IBM MQ classes for JMS attempt to use any of the JMS 2.0 API, the API method call fails with the exception `JMSCC5007`.

Related reference

Dependencies between properties of IBM MQ classes for JMS objects

TRANSPORT

PROVIDERVERSION unspecified

When the **PROVIDERVERSION** property of a connection factory is unspecified, the JMS client uses an algorithm to determine which mode of operation is used for connecting to the queue manager. A connection factory that was created in the JNDI namespace with a previous version of IBM MQ classes for JMS takes the unspecified value when the connection factory is used with the new version of IBM MQ classes for JMS.

If the **PROVIDERVERSION** property is unspecified, the algorithm is used when the `createConnection` method is called. The algorithm checks a number of connection factory properties to determine if IBM MQ messaging provider normal mode, normal mode with restrictions, or IBM MQ messaging provider migration mode is required. Normal mode is always attempted first, and then normal mode with restrictions. If neither of these types of connection can be made, the JMS client disconnects from the queue manager and then connects with the queue manager again to attempt a migration mode connection.

Checking of BROKERVER, BROKERQMGR, PSMODE, and BROKERCONQ properties

The checking of property values begins with the **BROKERVER** property as shown in Figure 1.

If the **BROKERVER** property is set to V1, the **TRANSPORT** property is checked next as shown in Figure 2. However, if the **BROKERVER** property is set to V2, the additional checking shown in Figure 1 is done before the **TRANSPORT** property is checked.

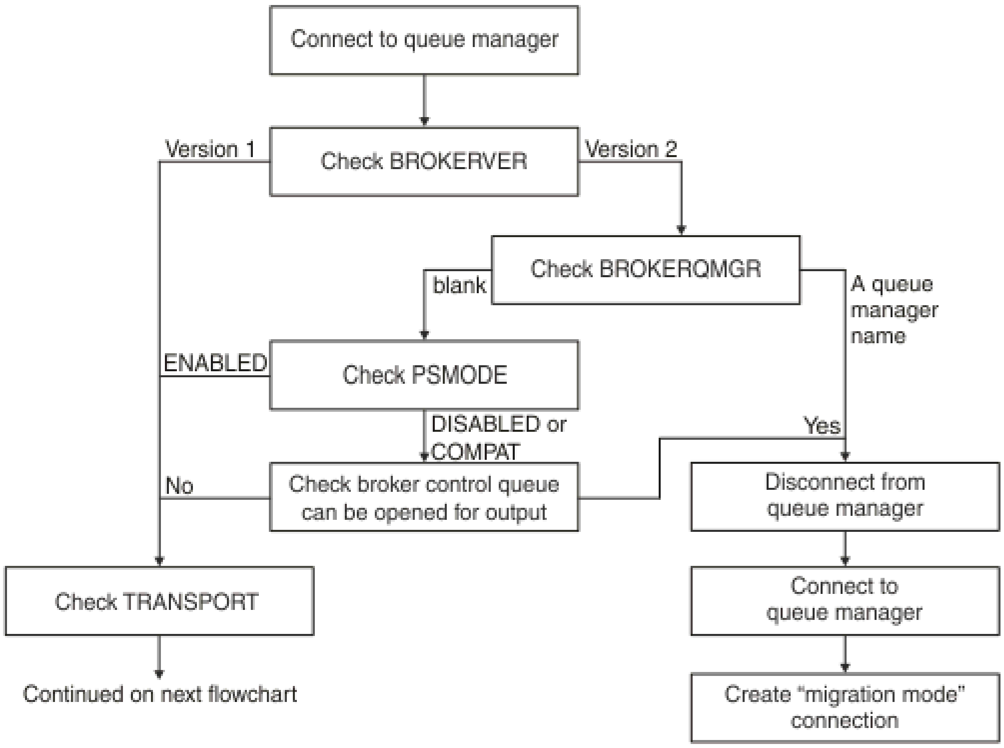

Figure 88. PROVIDERVERSION unspecified

If the **BROKERVER** property is set to V2, for a normal mode connection to be possible, the **BROKERQMGR** property must be blank. Additionally, either the **PSMODE** attribute on the queue manager must be set to ENABLED or the broker control queue specified by the **BROKERCONQ** property must not be able to be opened for output.

If the property values are set as required for a normal mode connection, checking next moves on to the **TRANSPORT** property as shown in Figure 2.

If the property values are not set as required for a normal mode connection, the JMS client disconnects from the queue manager and then reconnects and creates a migration mode connection. This happens in the following cases:

- If the **BROKERQMGR** property is blank and the **PSMODE** attribute on the queue manager is set to COMPAT or DISABLED and the broker control queue specified by the **BROKERCONQ** property can be opened for output (that is, MQOPEN for output succeeds).
- If the **BROKERQMGR** property specifies a queue name.

Checking of TRANSPORT property and command level

Figure 2 shows the checks that are made for the **TRANSPORT** property and command level of the queue manager.

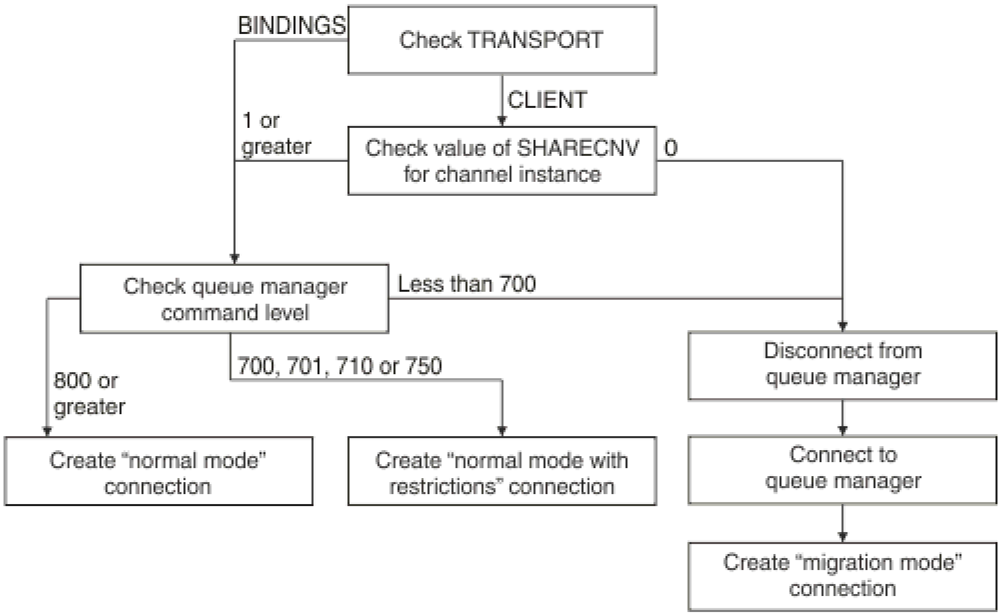

Figure 89. PROVIDERVERSION unspecified (continued)

A normal mode connection is created in either of the following cases:

- The **TRANSPORT** property of the connection factory is set to BINDINGS, and the queue manager has a command level of 800 or greater.
- The **TRANSPORT** property is set to CLIENT, the **SHARECNV** property on the server connection channel has a value of 1 or greater, and the queue manager has a command level of 800 or greater.

If the queue manager has a command level of 700, 701, 710 or 750, a normal mode with restrictions connection to the queue manager is created.

If the queue manager has a command level of less than 700, the JMS client disconnects from the queue manager and then connects with the queue manager again to create a migration mode connection.

A migration mode connection is also created if the **TRANSPORT** property is set to CLIENT and the **SHARECNV** property on the server connection channel has a value of 0.

Related reference

Dependencies between properties of IBM MQ classes for JMS objects
ALTER QMGR (PSMODE attribute)
BROKERCONQ
BROKERQMGR
BROKERVER
DEFINE CHANNEL (SHARECNV property)
TRANSPORT

When to override the PROVIDERVERSION default setting

If a connection factory that was created in the JNDI namespace with a previous version of IBM MQ classes for JMS is used with the new version of IBM MQ classes for JMS, the **PROVIDERVERSION** property for the connection factory is set to the default value of `unspecified` and an algorithm is used to determine which IBM MQ messaging provider mode of operation is used. However, there are two cases where you must override the default selection for the **PROVIDERVERSION** property so that the IBM MQ classes for JMS can work correctly.

Note: The migration mode that is described in this topic is for migration from IBM WebSphere MQ 6.0 to IBM WebSphere MQ 7.0. It does not apply to migration from later releases.

IBM WebSphere MQ 6.0, WebSphere Application Server Version 6.0.x, and WebSphere Message Broker Version 6 are out of support, and therefore this topic is included only for reference purposes.

When the **PROVIDERVERSION** property is set to the default of `unspecified`, an algorithm is used to determine which mode of operation to use, as described in "PROVIDERVERSION unspecified" on page 508. However, you cannot use this algorithm in the following two scenarios.

1. If WebSphere Message Broker and WebSphere Event Broker are in compatibility mode, you must specify a value for the **PROVIDERVERSION** property for WebSphere Message Broker and WebSphere Event Broker to work correctly.
2. If you are using WebSphere Application Server Version 6.0.1, WebSphere Application Server Version 6.0.2, or WebSphere Application Server Version 6.1, connection factories are defined by using the WebSphere Application Server administrative console.

   In WebSphere Application Server, the default value of the **BROKERVER** property on a connection factory is V2. The default value for the **BROKERVER** property for connection factories that are created by using the JMS administration tool **JMSAdmin** or MQ Explorer is V1. This property is now `unspecified` in IBM MQ.

If the **BROKERVER** property is set to V2, either because it was created by WebSphere Application Server or the connection factory has been used for publish/subscribe before, and has an existing queue manager that has a **BROKERCONQ** property defined (because it has been used for publish/subscribe messaging before), the IBM MQ messaging provider migration mode is used.

However, if you want the application to use peer-to-peer communication and the application is using an existing queue manager that has ever been used for publish/subscribe, and has a connection factory with **BROKERVER** set to 2, which is the default setting if the connection factory was created in WebSphere Application Server, the IBM MQ messaging provider migration mode is used. Using IBM MQ messaging provider migration mode in this case is unnecessary; use IBM MQ messaging provider normal mode instead. You can use one of the following methods to work around this:

- Set **BROKERVER** to 1 or unspecified. The option that you choose depends on your application.
- Set **PROVIDERVERSION** to 8, or 7, which are custom properties in WebSphere Application Server Version 6.1.

  Alternatively, use the client configuration property, or modify the queue manager connected so that it does not have the **BROKERCONQ** property set, or make the queue unusable.

Configuring provider version information in WebSphere Application Server

To configure provider version information in WebSphere Application Server, you can either use the administrative console or wsadmin commands.

Procedure

To configure provider version information for an IBM MQ connection factory or activation specification object in WebSphere Application Server, see the *Related information* for links to further information in the WebSphere Application Server product documentation.

Related information for WebSphere Application Server Version 8.5.5

IBM MQ messaging provider connection factory settings

`createWMQConnectionFactory` command

IBM MQ messaging provider activation specification settings

`createWMQActivationSpec` command

Related information for WebSphere Application Server Version 8.0.0

IBM MQ messaging provider connection factory settings

`createWMQConnectionFactory` command

IBM MQ activation specification settings

`createWMQActivationSpec` command

Related information for WebSphere Application Server Version 7.0.0
IBM MQ messaging provider connection factory settings
createWMQConnectionFactory command
IBM MQ activation specification settings
createWMQActivationSpec command

Remove WebSphere Application Server Version 7 and Version 8 durable subscriptions

When using the WebSphere MQ messaging provider with WebSphere Application Server Version 7 and Version 8, durable subscriptions created by message-driven bean applications bound to activation specifications are not removed. Durable subscriptions can be removed using either the MQ Explorer or a WebSphere MQ command line utility.

About this task

Message-driven bean applications running inside of WebSphere Application Server Version 7 and Version 8 that use WebSphere MQ messaging provider normal mode to connect to WebSphere MQ and take out a durable subscription can be configured to use either listener ports or activation specifications.

If the message-driven bean application is bound to a listener port, then the WebSphere MQ messaging provider will create the durable subscription for the application the first time the application is started. The durable subscription will be removed when the message-driven bean application is uninstalled from an application server, and the application server restarted.

Message-driven bean applications that are bound to an activation specification work in a slightly different way. The durable subscription is created for the application the first time the application is started. However, the durable subscription is not removed when the application is uninstalled and the application server restarted.

This can lead to a number of durable subscriptions remaining on a WebSphere MQ Publish/Subscribe engine for applications that are no longer installed in a WebSphere Application Server system. These subscriptions are known as "orphan subscriptions", and can lead to issues on the queue manager when the Publish/Subscribe engine is running.

When a message is published on a topic, the WebSphere MQ Publish/Subscribe engine makes a copy of that message for each durable subscription that is registered on that topic, and put it on an internal queue. The applications using that durable subscription will then pick up and consume the message from this internal queue.

If the message-driven bean application that was using that durable subscription is no longer installed, the copies of the published messages for the application will continue to be made. However, these messages will never be processed, which means that there could be a large number of messages remaining on the internal queue that will never be removed.

Before you begin

Subscriptions that are registered with the WebSphere MQ Publish/Subscribe engine will have a Subscription Name associated with them.

Durable subscriptions created by the WebSphere Application Server WebSphere MQ messaging provider for message-driven beans that are bound to activation specifications will have a Subscription Name in the following format:

```
JMS:<queue manager name>:<client identifier>:<subscription name>
```

Where:

<queue manager name>
: This is the name of the WebSphere MQ queue manager where the Publish/Subscribe engine is running.

<client identifier>
: This is the value of the Client ID property of the activation specification to which the message-driven bean is bound.

<subscription name>
: This is the value of the activation specification property Subscription name for the activation specification that the message-driven bean application has been configured to use.

For example, suppose that we have an activation specification that has been set up to connect to the queue manager testQM. The Specification has the following properties set:

- Client ID = testClientID
- Subscription name = durableSubscription1

If a message-driven bean that takes out a durable subscription is bound to this activation specification, a subscription is created on the WebSphere MQ publish/subscribe engine on the queue manager testQM that has the following Subscription name:

- JMS:testQM:testClientID:durableSubscription1

The subscriptions that have been registered with the WebSphere MQ publish/subscribe engine for a given queue manager can be viewed in one of the two following ways:

- The first option is to use the MQ Explorer. When the MQ Explorer has been connected to a queue manager that is being used for publish/subscribe work, the list of subscribers that are currently registered with the publish/subscribe engine can be viewed by clicking on the `IBM WebSphere MQ -><queue manager name>-> Subscriptions` entry in the navigation pane.
- The other way to view the subscriptions that have been registered with a publish/subscribe engine is to use the WebSphere MQ command line utility runmqsc and run the command **`display sub`**. To do this, bring up a command prompt, change to the <WebSphere MQ>\bin directory, and enter the following command to start runmqsc:
  - `runmqsc <queue manager name>`

  When the runmqsc utility has started, enter the following command to list all of the durable subscriptions currently registered with the publish/subscribe engine running on the queue manager to which runmqsc has connected:
  - `display sub(*) durable`

To check if the durable subscriptions registered with the publish/subscribe engines are still active:

1. Generate the list of durable subscriptions that have been registered with the Publish/Subscribe engine.
2. For each durable subscription:
   - Look at the Subscription name for the durable subscriber, and note the <client identifier> and <subscription name> value.
   - Look at the WebSphere Application Server systems that are connecting to this Publish/Subscribe engine. See if there are any activation specifications defined that have the Client ID property that matches the <client identifier> value and the Subscription name property that matches the <subscription name>.
   - If no activation specifications are found that have the Client ID and Subscription name properties that match the <client identifier> and <subscription name> fields in the WebSphere MQ subscription name, then there are no activation specifications using this durable subscription. The durable subscription can be deleted.
   - If there is an activation specification defined that matches the durable subscription name, then the final check that needs to be made is to see if there is a message-driven bean application using this activation specification. To do this:

- Make a note of the JNDI name for the activation specification that has taken out the durable subscription at which you are currently looking.
- Bring up the Configuration pane in the WebSphere Administrative Console for each message-driven bean application that has been installed.
- Click the Message Driven Bean listener bindings link in the Configuration pane.
- A table with information about the message-driven bean application is displayed. If the activation specification radio button is selected in the Bindings column, and the Target Resource JNDI name field contains the JNDI name for the activation specification that has taken out the durable subscription, then the subscription is still in use and cannot be deleted.
- If no message-driven bean applications can be found that are using the activation specification, then the durable subscription can be deleted.

Procedure

Once an "orphaned" durable subscription has been identified, it can be deleted using either the MQ Explorer or the WebSphere MQ command line utility **runmqsc**.

To delete an "orphaned" durable subscription using the MQ Explorer:

1. Highlight the entry for the subscription
2. Right click the entry, and select **Delete...** from the menu. A confirmation window will appear.
3. Check the subscription name displayed in the confirmation window is correct, and click **Yes**.

The MQ Explorer will now delete the subscription from the Publish/Subscribe engine, and clean up any internal resources associated with it (such as unprocessed messages that were published for the topic on which the durable subscription was registered).

To delete an "orphaned" durable subscription using the WebSphere MQ command line utility **runmqsc**, the command **delete sub** must be run:

1. Open a command prompt session
2. Navigate to the <WebSphere MQ>\bin directory
3. Enter the following command to start **runmqsc**:

   ```
   runmqsc <queue manager name>
   ```

4. When the **runmqsc** utility has started, enter:

   ```
   delete sub(<Subscription name>)
   ```

   where <*Subscription name*> is the subscription name of the durable subscription, which takes the form:

   - JMS:<*queue manager name*>:<*client identifier*>:<*subscription name*>

Configuring HP Integrity NonStop Server

Use this information to help you to configure your IBM MQ HP Integrity NonStop Server client for HP Integrity NonStop Server installation.

For details about configuring a client by using a configuration file, see “Configuring a client using a configuration file” on page 47.

For details about configuring a client by using environment variables, see “Using IBM MQ environment variables” on page 71.

If you are performing IBM MQ HP Integrity NonStop Server client for HP Integrity NonStop Server operations under TMF/Gateway, see the subtopics for information about how to configure the TMF/Gateway. Included are an overview of the Gateway process, configuring the Gateway to run under

Pathway, and configuring the client initialization file to enable your IBM MQ HP Integrity NonStop Server client for HP Integrity NonStop Server to reach the TMF Gateway.

This section also contains IBM MQ HP Integrity NonStop Server client for HP Integrity NonStop Server specific information about granting permissions to channels.

Gateway process overview

The HP NonStop Transaction Management Facility (TMF) provides services to enable a gateway process to register as a resource manager. The IBM WebSphere MQ for HP Integrity NonStop Server provided TMF/Gateway process runs under Pathway.

IBM WebSphere MQ for HP Integrity NonStop Server registers a single gateway process for each queue manager that is coordinated by TMF, therefore you must configure a separate TMF/Gateway for each queue manager that is to participate in TMF coordinated units of work. This registration is so that each queue manager is an independent resource manager, and for administrative purposes, registering each queue manager once with HP NonStop TMF results in an easy to understand mapping.

For multiple installations of IBM WebSphere MQ for HP Integrity NonStop Server, you must nominate a single gateway process from one of these installations for each queue manager to be coordinated by TMF.

The interface to the gateway process supports any client at the same version or earlier.

For more information about administering the gateway process, see Administering HP Integrity NonStop Server .

Configuring Gateway to run under Pathway

TMF/Gateway is the interface between the HP NonStop Transaction Management Facility (TMF) and IBM MQ that enables TMF to be the transaction coordinator for IBM MQ transactions.

The IBM MQ provided TMF/Gateway converts transactions from TMF coordination into eXtended Architecture (XA) transaction coordination to communicate with the remote queue manager.

You must have one TMF/Gateway per queue manager that requires coordination, and client configuration is required so that the client can connect to the correct Gateway.

The TMF/Gateway can use all the mechanisms available to the client to communicate with a queue manager. Configure the TMF/Gateway in the way you would for your other applications.

The TMF/Gateway is not a HP Integrity NonStop Server process pair and is designed to run in a Pathway environment. The TMF/Gateway creates permanent resources within TMF, which it reuses on subsequent runs, therefore the TMF/Gateway must always be run under the same user authority.

Defining the serverclass

TMF/Gateway is hosted as a serverclass within a Pathway environment. To define the serverclass, you must set the following server attributes:

PROCESSTYPE = OSS
: Specifies the type of servers in the serverclass. The Gateway process is a multi-threaded OSS program. This attribute is mandatory, and must be set to OSS.

MAXSERVERS = 1
: Specifies the maximum number of server processes in this serverclass that can run at the same time. There can be only a single Gateway process for any queue manager. This attribute is mandatory and must be set to 1.

NUMSTATIC = 1
: Specifies the maximum number of static servers within this serverclass. The Gateway process must be run as a static server. This attribute is mandatory and must be set to 1.

TMF = ON
: Specifies whether servers in this serverclass can lock and update data files that are audited by the TMF subsystem. The Gateway process participates in the TMF transactions of IBM MQ client applications therefore this attribute must be set to `ON`.

PROGRAM = <OSS installation path>/opt/mqm/bin/runmqtmf
: For IBM MQ client for IBM MQ, this attribute must be `runmqtmf`. This attribute must be the absolute OSS path name. Case is significant.

ARGLIST = *-m<QMgr name>* [,-c<channel name>][,-p<port>][,-h<host name>][,-n<max threads>]
: These attributes provide parameters to the Gateway process, where:

- `QMgrName` is the name of the queue manager for this Gateway process. If you are using a queue sharing group (or other port distribution technology), this parameter must be targeted to a specific queue manager. This parameter is mandatory.
- `channel name` is the name of the server channel on the queue manager to be used by the Gateway process. This parameter is optional.
- `port` is the TCP/IP port for the queue manager. This parameter is optional.
- `host name` is the host name for the queue manager. This parameter is optional.
- `max threads` is the maximum number of worker threads that are created by the Gateway process. This parameter can be a value of 10 or greater. The lowest value that is used is 10 even if a value lower than 10 is specified. If no value is provided, the Gateway process creates up to a maximum of 50 threads.

Use the `-c`, `-p`, and `-h` attributes as an alternative method of providing connection information to the Gateway, in addition to that described in "Configuring the TMF/Gateway using environment variables" on page 516. If you specify one or more, but not all of the `-c`, `-p`, and `-h` attributes, then those attributes that you do not specify default to the following values:

- `channel name` defaults to `SYSTEM.DEF.SVRCONN`
- `host name` defaults to `localhost`
- `port` defaults to 1414

If any of the parameters you supply are invalid, the TMF/Gateway issues diagnostic message AMQ5379 to the error log and terminates.

OWNER = ID
: The user ID under which the Gateway runs and that must be granted connect authority to the queue manager.

SECURITY = "value"
: Specifies the users, in relation to the `Owner` attribute, who can access the Gateway from an IBM MQ client application.

HIGHPIN = ON
: Optionally specifies whether the Gateway process can run as a high pin process.

CWD = <OSS_install_path>/var/mqm
: Specifies the working directory for the Gateway process.

`LINKDEPTH` and `MAXLINKS` must be configured with values appropriate for the expected number of IBM MQ client applications that might want to concurrently communicate with the Gateway. If these values are set too low, you might see occurrences of the error message AMQ5399 issued from client applications.

For more information about these server attributes, see the *HP NonStop TS/MP 2.5 System Management Manual*.

Configuring the TMF/Gateway using environment variables

One of the most commonly used methods to define the TMF/Gateway is to set the MQSERVER environment variable, for example:

```
ENV MQSERVER=<channel name>/<transport>/<host name>(<listener port>)
```

ENV at the beginning of the command is Pathway notation.

Configuring the client initialization file

If you are using the HP NonStop Transaction Management Facility (TMF), you must have an IBM MQ client initialization file to enable your IBM MQ client for the HP Integrity NonStop Server to reach the TMF Gateway.

An IBM MQ client initialization file for HP Integrity NonStop Server can be held in a number of locations, for more information, see "Location of the client configuration file" on page 49.

For details of the contents of the configuration file, together with an example, see "Configuring a client using a configuration file" on page 47. Use the TMF and TmfGateway stanzas to specify the TMF queue manager and server details, for more information, see "TMF and TmfGateway stanzas" on page 70.

An example of the entries for a IBM MQ client for HP Integrity NonStop Server is:

```
TMF:
PathMon=$PSD1P

TmfGateway:
QManager=MQ5B
Server=MQ-MQ5B

TmfGateway:
QManager=MQ5C
Server=MQ-MQ5C
```

For more information about configuring a client using environment variables, see "Using IBM MQ environment variables" on page 71.

Granting permissions to channels

Granting permissions to channels on IBM MQ client for HP Integrity NonStop Server is identical to other operating systems, however you must know the identification of the owner that the gateway is running under.

You can then use the identification of the owner of the gateway to grant appropriate permissions. The important difference is that granting permissions to queue manager channels is not under the authority of any application.

Use the setmqaut command to both to grant an authorization, that is, give an IBM MQ principal or user group permission to perform an operation, and to revoke an authorization, that is, remove the permission to perform an operation.

V 8.0.0.4 Configuring IBM MQ using Docker

Use this information to configure IBM MQ using Docker.

Docker allows you to package an IBM MQ queue manager or IBM MQ client application, with all of its dependencies, into a standardized unit for software development.

Changes to your application can be deployed to test and staging systems quickly and easily. This feature can be a major benefit to continuous delivery in your enterprise.

V 8.0.0.4 Docker support on Linux systems

Note the following information if you are using Docker on a Linux system:

- The base image used by the Docker image must use a Linux operating system that is supported.

- The host image, that the Docker container is running on, must use a 3.16 or newer Linux kernel, and must be an operating system supported by IBM MQ.
- You must use the IBM MQ installers to install the product inside the Docker image.
- Only the following packages are supported:
  - `MQSeriesRuntime`
  - `MQSeriesServer`
  - `MQSeriesClient`
  - `MQSeriesJava`
  - `MQSeriesJRE`
  - `MQSeriesGSKit`
  - `MQSeriesMsg`
  - `MQSeriesMan`
- The queue manager data directory (`/var/mqm` by default) must be stored on a Docker volume which keeps persistent state.

  Important: You cannot use the union file system.

  You must either mount a host directory as a data volume, or use a data volume container. See Manage data in containers for more information.
- Running the mqconfig command inside the container must pass, to ensure that the container has the correct configuration.
- Applications can only be locally bound to the queue manager (client BINDINGS mode) when running in the same container as that queue manager.
- You must be able to run IBM MQ control commands, such as **`endmqm`**, within the container.
- You must be able to obtain files and directories from within the container for diagnostic purposes.

V 8.0.0.4 Planning your own IBM MQ queue manager image using Docker

Use this information to configure IBM MQ using Docker. There are several requirements to consider when running an IBM MQ queue manager in Docker. The sample Docker image provides a way to handle these requirements, but if you want to use your own image, you need to consider how these requirements are handled.

Process supervision

When you run a Docker container, you are essentially running a single process (PID 1 inside the container), which can later spawn child processes.

If the main process ends, Docker stops the container. An IBM MQ queue manager requires multiple processes to be running in the background.

For this reason, you need to make sure that your main process stays active as long as the queue manager is running. It is good practice to check that the queue manager is active from this process, for example, by performing administrative queries.

Populating `/var/mqm`

Docker containers must be configured with `/var/mqm` as a Docker volume.

When you do this, the directory of the volume is empty when the container first starts. This directory is usually populated at installation time, but installation and runtime are separate environments when using Docker.

To solve this, when your container starts, you can use the **`amqicdir`** command to populate `/var/mqm` when it runs for the first time.

V 8.0.0.4 Building a sample IBM MQ queue manager image using Docker

Use this information to build a sample Docker image for running an IBM MQ queue manager in a Docker container.

Firstly, you will build a base image containing an Ubuntu Linux file system and a clean installation of IBM MQ.

Secondly, you will build another Docker image layer on top of the base, which adds some IBM MQ configuration to allow basic user ID and password security.

Finally, you will run a Docker container using this image as its file system, with the contents of /var/mqm provided by a container-specific Docker volume on the host file system of Docker.

V 8.0.0.4 Building a sample base IBM MQ queue manager image

In order to use IBM MQ in Docker, you need initially to build a base image with a clean IBM MQ installation. The following steps show you how to build a sample base image, using code hosted on GitHub.

Procedure

1. Install the prerequisite packages.

   These instructions make use of some Linux packages that you must install.

   - On Ubuntu:

     ```
     sudo apt-get install python git
     ```

   - On Red Hat Enterprise Linux:

     ```
     sudo yum install python git
     ```

2. Create a `downloads` directory by issuing the command `mkdir downloads`.
3. Download the IBM MQ server for Linux image, using Passport Advantage®.

   See Installation using Electronic Software Download for more details.

   For example, select the `WS_MQ_V8.0.0.4_LINUX_ON_X86_64_IM.tar.gz` file, and place the file in the `downloads` directory that you have created.
4. Make the IBM MQ server for Linux image (`tar.gz`) file available on an HTTP or FTP server.

   The reason for this is to save space in the Docker image layers. Every instruction in a Docker file causes a new image layer to be created.

   If you use the **ADD** or **COPY** instructions, followed by a **RUN** instruction to install, then the files added or copied will be committed to a new image layer.

   Even if you delete the file in subsequent layers, the file still exists in the previous layer. For this reason, it is good practice to download and install within a single **RUN** command, which means the files need to be available on the network.

   For example, you can use Python to run an HTTP server, serving all files in your current directory:

```
pushd downloads
nohup python -m SimpleHTTPServer 8000 &
popd
```

5. Extract the sample files, for building a supported Docker image, from GitHub by issuing the following command:

```
git clone -b mq-8 https://github.com/ibm-messaging/mq-docker mq-docker
```

6. Identify your local IP address.

Your address is specific to your local environment, but should be available if you run the following command:

```
ip addr show
```

Note that `localhost` does not work.

7. Edit the Docker file and change the existing entry for IBM MQ_URL to point to the file on your local file server.

   For example:

```
MQ_URL=http://10.0.2.15:8000/WS_MQ_V8.0.0.4_LINUX_ON_X86_64_IM.tar.gz
```

8. Change your directory by issuing the following command:

```
cd mq-docker
```

9. Build the base IBM MQ image by issuing the following command:.

```
sudo docker build --tag mq .
```

Results

You now have a base Docker image with IBM MQ installed.

V 8.0.0.4 Building a sample configured IBM MQ queue manager image

Once you have built your generic base IBM MQ Docker image, you need to apply your own configuration to allow secure access. To do this, create your own Docker image, using the generic image as a parent. The following steps show you how to build a sample image, with a minimal security configuration.

Procedure

1. Create a new directory, and add a file called `config.mqsc`, with the following contents:

```
DEFINE CHANNEL(PASSWORD.SVRCONN) CHLTYPE(SVRCONN)
SET CHLAUTH(PASSWORD.SVRCONN) TYPE(BLOCKUSER) USERLIST('nobody') +
DESCR('Allow privileged users on this channel')
SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) DESCR('BackStop rule')
SET CHLAUTH(PASSWORD.SVRCONN) TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL) CHCKCLNT(REQUIRED)
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) ADOPTCTX(YES)
REFRESH SECURITY TYPE(CONNAUTH)
```

   Note that the preceding example uses simple user ID and password authentication. However, you can apply any security configuration that your enterprise requires.

2. Create a file called `Dockerfile`, with the following contents:

```
FROM mq
RUN useradd johndoe -G mqm && \
    echo johndoe:passw0rd | chpasswd
COPY config.mqsc /etc/mqm/
```

   where:

   - `johndoe` is the user ID that you want to add
   - `passw0rd` is the original password

3. Build your custom Docker image using the following command:

```
sudo docker build -t mymq .
```

   where "`.`" is the directory containing the two files you have just created.

   Docker then creates a temporary container using that image, and runs the remaining commands.

The **RUN** command adds a user named `johndoe` with password `passw0rd` and the **COPY** command adds the `config.mqsc` file into a specific location known by the parent image.

4. Run your new customized image to create a new container, with the disk image you have just created.

   Your new image layer did not specify any particular command to run, so that has been inherited from the parent image. The entry point of the parent (the code is available on GitHub):

   - Creates a queue manager
   - Starts the queue manager
   - Creates a default listener
   - Then runs any MQSC commands from `/etc/mqm/config.mqsc`.

   Issue the following commands to run your new customized image:

```
sudo docker run \
  --env LICENSE=accept \
  --env MQ_QMGR_NAME=QM1 \
  --volume /var/example:/var/mqm \
  --publish 1414:1414 \
  --detach \
  mymq
```

   where the:

   First env parameter
   Passes an environment variable into the container, which acknowledges your acceptance of the license for IBM IBM WebSphere MQ. You can also set the LICENSE variable to view to view the license.

   See IBM MQ license information for further details on IBM MQ licenses.

   Second env parameter
   Sets the queue manager name that you are using.

   Volume parameter
   Tells the container that whatever MQ writes to `/var/mqm` should actually be written to `/var/example` on the host.

   This option means that you can easily delete the container later, and still keep any persistent data. This option also makes it easier to view log files.

   Publish parameter
   Maps ports on the host system to ports in the container. The container runs by default with its own internal IP address, which means that you need to specifically map any ports that you want to expose.

   In this example, that means mapping port 1414 on the host to port 1414 in the container.

   Detach parameter
   Runs the container in the background.

Results

You have built a configured docker image and can view running containers using the docker **ps** command. You can view the IBM MQ processes running in your container using the docker **top** command.

Attention: If your container is not shown when you use the docker **ps** command the container might have failed. You can see failed containers using the command docker **ps -a**.

The container ID will be shown by using the docker **ps -a** command, and was also printed when you issued the docker **run** command.

You can view the logs of a container using the docker **logs ${CONTAINER_ID}** command.

A common problem is that **mqconfig** indicates that certain kernel settings on the Docker host are not correct. Kernel settings are shared between the Docker host and containers, and need to be set correctly (see Hardware and software requirements on UNIX and Linux systems.

For example, the maximum number of open files can be set using the command **`sysctl fs.file-max=524288`**.

z/OS Configuring queue managers on z/OS

Use these instructions to configure queue managers on IBM MQ for z/OS.

Before you configure IBM MQ, read about the IBM MQ for z/OS concepts in IBM MQ for z/OS concepts.

z/OS Read about how to plan your IBM MQ for z/OS environment in Planning your IBM MQ environment on z/OS .

Related concepts

"Creating and managing queue managers on distributed platforms" on page 5
Before you can use messages and queues, you must create and start at least one queue manager and its associated objects.

IBM MQ technical overview

"Configuring" on page 5
Create one or more queue managers on one or more computers, and configure them on your development, test, and production systems to process messages that contain your business data.

Security

"Configuring distributed queuing" on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

"Configuring connections between the client and server" on page 14
To configure the communication links between IBM MQ MQI clients and servers, decide on your communication protocol, define the connections at both ends of the link, start a listener, and define channels.

z/OS The IBM MQ for z/OS utilities

Related tasks

Administering IBM MQ

z/OS Administering IBM MQ for z/OS

Planning

z/OS Issuing commands

Preparing to customize your IBM MQ for z/OS queue managers

Use this topic when customizing your queue managers with details of installable features, national language features, and information about testing, and setting up security.

Preparing for customization

The Program Directory for WebSphere MQ for z/OS lists the contents of the IBM MQ installation tape, the program and service level information for IBM MQ, and describes how to install IBM MQ for z/OS using the System Modification Program Extended (SMP/E).

When you have installed IBM MQ, you must carry out a number of tasks before you can make it available to users. See the following sections for a description of these tasks:

- "Customizing IBM MQ for z/OS" on page 526
- "Testing your queue manager on z/OS" on page 579
- Setting up security on z/OS

If you are migrating from a previous version of IBM MQ for z/OS, you do not need to perform most of the customization tasks. See Migrating and upgrading IBM MQ for more information about the tasks you must perform.

Installable features of IBM MQ for z/OS

IBM MQ for z/OS comprises the following features:

Base
This is required; it comprises all the main functions, including

- Administration and utilities
- Support for CICS, IMS, and batch type applications using the IBM MQ Application Programming Interface, or C++
- Distributed queuing facility (supporting both TCP/IP and APPC communications)

National language features
These contain error messages and panels in all the supported national languages. Each language has a language letter associated with it. The languages and letters are:

C
Simplified Chinese

E
U.S. English (mixed case)

F
French

K
Japanese

U
U.S. English (uppercase)

You must install the US English (mixed case) option. You can also install one or more other languages. (The installation process for other languages requires US English (mixed case) to be installed, even if you are not going to use US English (mixed case).)

IBM MQ for z/OS Unix System Services Components
This feature is optional. Select this feature if you want to build and run Java applications that use the Java Message Service (JMS) to connect to IBM MQ for z/OS or if you want to build and run HTTP applications which use HTTP to connect to IBM MQ for z/OS.

Libraries that exist after installation

IBM MQ is supplied with a number of separate load libraries. Table 35 on page 523 shows the libraries that might exist after you have installed IBM MQ.

Table 35. IBM MQ libraries that exist after installation

Name	Description
thlqual.SCSQANLC	Contains the load modules for the Simplified Chinese version of IBM MQ.
thlqual.SCSQANLE	Contains the load modules for the U.S. English (mixed case) version of IBM MQ.
thlqual.SCSQANLF	Contains the load modules for the French version of IBM MQ.
thlqual.SCSQANLK	Contains the load modules for the Japanese version of IBM MQ.
thlqual.SCSQANLU	Contains the load modules for the U.S. English (uppercase) version of IBM MQ.
thlqual.SCSQASMS	Contains source for assembler sample programs.

Table 35. IBM MQ libraries that exist after installation (continued)	
Name	**Description**
thlqual.SCSQAUTH	The main repository for all IBM MQ product load modules; it also contains the default parameter module, CSQZPARM. This library must be APF-authorized and in PDS-E format.
thlqual.SCSQCICS	Contains extra load modules that must be included in the CICS DFHRPL concatenation. This library must be APF-authorized and in PDS-E format.
thlqual.SCSQCLST	Contains CLISTs used by the sample programs.
thlqual.SCSQCOBC	Contains COBOL copybooks, including copybooks required for the sample programs.
thlqual.SCSQCOBS	Contains source for COBOL sample programs.
thlqual.SCSQCPPS	Contains source for C++ sample programs.
thlqual.SCSQC37S	Contains source for C sample programs.
thlqual.SCSQC370	Contains C headers, including headers required for the sample programs.
thlqual.SCSQDEFS	Contains side definitions for C++ and the Db2 DBRMs for shared queuing.
thlqual.SCSQEXEC	Contains REXX executable files to be included in the SYSEXEC or SYSPROC concatenation if you are using the IBM MQ operations and control panels.
thlqual.SCSQHPPS	Contains header files for C++.
thlqual.SCSQINST	Contains JCL for installation jobs.
thlqual.SCSQLINK	Early code library. Contains the load modules that are loaded at system initial program load (IPL). The library must be APF-authorized.
thlqual.SCSQLOAD	Load library. Contains load modules for non-APF code, user exits, utilities, samples, installation verification programs, and adapter stubs. The library does not need to be APF-authorized and does not need to be in the link list. This library must be in PDS-E format.
thlqual.SCSQMACS	Contains Assembler macros including: sample macros, product macros, and system parameter macros.
thlqual.SCSQMAPS	Contains CICS mapsets used by sample programs.
thlqual.SCSQMSGC	Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the Simplified Chinese language feature for the IBM MQ operations and control panels.
thlqual.SCSQMSGE	Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the U.S. English (mixed case) language feature for the IBM MQ operations and control panels.
thlqual.SCSQMSGF	Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the French language feature for the IBM MQ operations and control panels.

Table 35. IBM MQ libraries that exist after installation (continued)	
Name	**Description**
thlqual.SCSQMSGK	Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the Japanese language feature for the IBM MQ operations and control panels.
thlqual.SCSQMSGU	Contains ISPF messages to be included in the ISPMLIB concatenation if you are using the U.S. English (uppercase) language feature for the IBM MQ operations and control panels.
thlqual.SCSQMVR1	Contains the load modules for distributed queuing. This library must be APF-authorized and in PDS-E format.
thlqual.SCSQPLIC	Contains PL/I include files.
thlqual.SCSQPLIS	Contains source for PL/I sample programs.
thlqual.SCSQPNLA	Contains IPCS panels, for the dump formatter, to be included in the ISPPLIB concatenation. Also contains panels for IBM MQ sample programs.
thlqual.SCSQPNLC	Contains ISPF panels to be included in the ISPPLIB concatenation if you are using the Simplified Chinese language feature for the IBM MQ operations and control panels.
thlqual.SCSQPNLE	Contains ISPF panels to be included in the ISPPLIB concatenation if you are using the U.S. English (mixed case) language feature for the IBM MQ operations and control panels.
thlqual.SCSQPNLF	Contains ISPF panels to be included in the ISPPLIB concatenation if you are using the French language feature for the IBM MQ operations and control panels.
thlqual.SCSQPNLK	Contains ISPF panels to be included in the ISPPLIB concatenation if you are using the Japanese language feature for the IBM MQ operations and control panels.
thlqual.SCSQPNLU	Contains ISPF panels to be included in the ISPPLIB concatenation if you are using the U.S. English (uppercase) language feature for the IBM MQ operations and control panels.
thlqual.SCSQPROC	Contains sample JCL and default system initialization data sets.
thlqual.SCSQSNLC	Contains the load modules for the Simplified Chinese versions of the IBM MQ modules that are required for special purpose function (for example the early code).
thlqual.SCSQSNLE	Contains the load modules for the U.S. English (mixed case) versions of the IBM MQ modules that are required for special purpose function (for example the early code).
thlqual.SCSQSNLF	Contains the load modules for the French versions of the IBM MQ modules that are required for special purpose function (for example the early code).
thlqual.SCSQSNLK	Contains the load modules for the Japanese versions of the IBM MQ modules that are required for special purpose function (for example the early code).
thlqual.SCSQSNLU	Contains the load modules for the U.S. English (uppercase) versions of the IBM MQ modules that are required for special purpose function (for example the early code).

Table 35. IBM MQ libraries that exist after installation (continued)	
Name	**Description**
thlqual.SCSQTBLC	Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the Simplified Chinese language feature for the IBM MQ operations and control panels.
thlqual.SCSQTBLE	Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the U.S. English (mixed case) language feature for the IBM MQ operations and control panels.
thlqual.SCSQTBLF	Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the French language feature for the IBM MQ operations and control panels.
thlqual.SCSQTBLK	Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the Japanese language feature for the IBM MQ operations and control panels.
thlqual.SCSQTBLU	Contains ISPF tables to be included in the ISPTLIB concatenation if you are using the U.S. English (uppercase) language feature for the IBM MQ operations and control panels.

Note: Do not modify or customize any of these libraries. If you want to make changes, copy the libraries and make your changes to the copies.

Related concepts

IBM MQ for z/OS concepts

“Setting up communications with other queue managers” on page 587
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

“Using IBM MQ with IMS” on page 616
The IBM MQ -IMS adapter, and the IBM MQ - IMS bridge are the two components which allow IBM MQ to interact with IMS.

“Using IBM MQ with CICS” on page 624
To use IBM MQ with CICS, you must configure the IBM MQ CICS adapter and, optionally, the IBM MQ CICS bridge components.

“Using OTMA exits in IMS” on page 626
Use this topic if you want to use IMS Open Transaction Manager Access exits with IBM MQ for z/OS.

Related tasks

Administering IBM MQ for z/OS

Related reference

“Upgrading and applying service to Language Environment or z/OS Callable Services” on page 624
The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

Customizing IBM MQ for z/OS

Use this topic as a step by step guide for customizing your IBM MQ system.

This topic leads you through the various stages of customizing IBM MQ after you have successfully installed it. The installation process is described in the Program Directory, available to download from the IBM Publications Center.

Samples are supplied with IBM MQ to help you with your customization. The sample data set members have names beginning with the four characters CSQ4 and are in the library thlqual.SCSQPROC.

Before you perform the customization tasks described in this topic, there are a number of configuration options that you must consider because they affect the performance and resource requirements of IBM MQ for z/OS. For example, you must decide which globalization libraries you want to use.

Configuration options

For more information about these options, see Planning on z/OS.

The description of each task in this section indicates whether:

- The task is part of the process of customizing IBM MQ. That is, you perform the task once when you customize IBM MQ on the z/OS system. (In a parallel sysplex, you must perform the task for each z/OS system in the sysplex, and ensure that each z/OS system is set up identically.)
- The task is part of adding a queue manager. That is, you perform the task once for each queue manager when you add that queue manager.
- You need to perform the task when migrating. If you are migrating from a previous version of IBM MQ for z/OS, you might not need to perform all these tasks.

Review the tasks when you apply corrective maintenance to IBM MQ and when you install a new version or release of IBM MQ.

None of the tasks require you to perform an IPL of your z/OS system, if you use commands to change the various z/OS system parameters, and perform “Task 12: Update SYS1.PARMLIB members” on page 543 as suggested.

To simplify operations and to aid with problem determination, ensure that all z/OS systems in a sysplex are set up identically, so that queue managers can be quickly created on any system in an emergency.

For ease of maintenance, consider defining aliases to refer to your IBM MQ libraries; for more information, see Using an alias to refer to an IBM MQ library.

Identify the national language support libraries

You need to specify the appropriate globalization libraries in the JCL that you want to use with IBM MQ (as described in the following sections). Each language is identified by a language letter:

C
Simplified Chinese

E
U.S. English (mixed case)

F
French

K
Japanese

U
U.S. English (uppercase)

Table 36. National language feature libraries

Description	Japanese	Simplified Chinese	U.S. English (mixed case)	U.S. English (uppercase)	French
Load modules	thlqual.SCSQANLK	thlqual.SCSQANLC	thlqual.SCSQANLE	thlqual.SCSQANLU	thlqual.SCSQANLF
ISPF messages	thlqual.SCSQMSGK	thlqual.SCSQMSGC	thlqual.SCSQMSGE	thlqual.SCSQMSGU	thlqual.SCSQMSGF
ISPF panels	thlqual.SCSQPNLK	thlqual.SCSQPNLC	thlqual.SCSQPNLE	thlqual.SCSQPNLU	thlqual.SCSQPNLF

Description	Japanese	Simplified Chinese	U.S. English (mixed case)	U.S. English (uppercase)	French
Special purpose function (for example, early code)	thlqual.SCSQSNLK	thlqual.SCSQSNLC	thlqual.SCSQSNLE	thlqual.SCSQSNLU	thlqual.SCSQSNLF
ISPF tables	thlqual.SCSQTBLK	thlqual.SCSQTBLC	thlqual.SCSQTBLE	thlqual.SCSQTBLU	thlqual.SCSQTBLF

Table 36. National language feature libraries (continued)

Customization summary

The following table lists all the steps required to customize IBM MQ for z/OS. It also indicates the following:

- Whether the step has to be performed once only, or repeated for each queue manager.
- Whether you need to repeat the step for each queue-sharing group, or omit the step if you are not using queue-sharing groups.
- Whether the step is required if you are migrating from a previous version of IBM MQ. Some steps might be needed, depending on what you decide about data set and queue manager names; these steps are marked 'Review'.

Table 37. Customization summary

Task	Required when migrating	Repeat for each queue manager	Queue-sharing groups
"Task 1: Identify the z/OS system parameters" on page 530	Review	-	-
"Task 2: APF authorize the IBM MQ load libraries" on page 530	Review	-	-
"Task 3: Update the z/OS link list and LPA" on page 531	Review	-	-
"Task 4: Update the z/OS program properties table" on page 533	-	-	-
"Task 5: Define the IBM MQ subsystem to z/OS" on page 533	-	X	-
"Task 6: Create procedures for the IBM MQ queue manager" on page 537	Review	X	-
"Task 7: Create procedures for the channel initiator" on page 538	Review	X	-
"Task 8: Define the IBM MQ subsystem to a z/OS WLM service class" on page 539	-	X	-
"Task 9: Set up the Db2 environment" on page 539	Review	-	Omit if not used
"Task 10: Set up the coupling facility" on page 541	Review	-	Repeat for each
"Task 11: Implement your ESM security controls" on page 542	Review	X	X

Table 37. Customization summary (continued)			
Task	**Required when migrating**	**Repeat for each queue manager**	**Queue-sharing groups**
"Task 12: Update SYS1.PARMLIB members" on page 543	Review	-	-
"Task 13: Customize the initialization input data sets" on page 543	X	X	-
"Task 14: Create the bootstrap and log data sets" on page 546	-	X	-
"Task 15: Define your page sets" on page 547	-	X	-
"Task 16: Add the IBM MQ entries to the Db2 tables" on page 548	Review	X	Repeat for each
"Task 17: Tailor your system parameter module" on page 548	X	X	-
"Task 18: Tailor the channel initiator parameters" on page 570	X	X	-
"Task 19: Set up Batch, TSO, and RRS adapters" on page 572	Review	-	-
"Task 20: Set up the operations and control panels" on page 572	Review	-	-
"Task 21: Include the IBM MQ dump formatting member" on page 574	X	-	-
"Task 22: Suppress information messages" on page 574	-	-	-
"Task 23: Create procedures for Advanced Message Security" on page 575	Review	X	-
"Task 24: Set up the started task user Advanced Message Security" on page 576	Review	X	-
"Task 25: Grant RACDCERT permissions to the security administrator for Advanced Message Security" on page 578	-	-	-
"Task 26: Grant users resource permissions for IBM MQ Advanced Message Security" on page 578	-	-	-

Related concepts

IBM MQ for z/OS concepts

"Setting up communications with other queue managers" on page 587

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

"Using IBM MQ with IMS" on page 616

The IBM MQ -IMS adapter, and the IBM MQ - IMS bridge are the two components which allow IBM MQ to interact with IMS.

"Using IBM MQ with CICS" on page 624

To use IBM MQ with CICS, you must configure the IBM MQ CICS adapter and, optionally, the IBM MQ CICS bridge components.

"Using OTMA exits in IMS" on page 626
Use this topic if you want to use IMS Open Transaction Manager Access exits with IBM MQ for z/OS.

Related tasks

Administering IBM MQ for z/OS

Related reference

"Upgrading and applying service to Language Environment or z/OS Callable Services" on page 624
The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

Related information

Program Directory for IBM MQ for z/OS

Task 1: Identify the z/OS system parameters

Some of the tasks involve updating the z/OS system parameters. You need to know which ones were specified when the system IPL was performed.

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *You might need to perform this task when migrating from a previous version.*

SYS1.PARMLIB(IEASYSpp) contains a list of parameters that point to other members of SYS1.PARMLIB (where pp represents the z/OS system parameter list that was used to perform an IPL of the system).

The entries you need to find are:

For "Task 2: APF authorize the IBM MQ load libraries" on page 530:
PROG=xx or APF=aa point to the Authorized Program Facility (APF) authorized library list (member PROGxx or IEFAPFaa)

For "Task 3: Update the z/OS link list and LPA" on page 531:
LNK=kk points to the link list (member LNKLSTkk) LPA=mm points to the LPA list (member LPALSTmm)

For "Task 4: Update the z/OS program properties table" on page 533:
SCH=xx points to the Program Properties Table (PPT) (member SCHEDxx)

For "Task 5: Define the IBM MQ subsystem to z/OS" on page 533:
SSN=ss points to the defined subsystem list (member IEFSSNss)

Task 2: APF authorize the IBM MQ load libraries

APF-authorize various libraries. Some load modules might already be authorized.

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *If you are using queue-sharing groups, you must ensure that the settings for IBM MQ are identical on each z/OS system in the sysplex.*
- *You might need to perform this task when migrating from a previous version.*

The IBM MQ load libraries thlqual.SCSQAUTH and thlqual.SCSQLINK must be APF-authorized. You must also APF-authorize the libraries for your national language feature (thlqual.SCSQANLx and thlqual.SCSQSNLx) and for the distributed queuing feature (thlqual.SCSQMVR1). If you are using IBM MQ Advanced Message Security you must also APF authorize the library thlqual.SDRQAUTH.

However, all load modules in the LPA are automatically APF-authorized. So are all members of the link list if the SYS1.PARMLIB member IEASYSpp contains the statement:

```
LNKAUTH=LNKLST
```

LNKAUTH=LNKLST is the default if LNKAUTH is not specified.

Depending on what you choose to put in the LPA or linklist (see "Task 3: Update the z/OS link list and LPA" on page 531), you might not need to put the libraries in the APF link list

Note: You must APF-authorize all the libraries that you include in the IBM MQ STEPLIB. If you put a library that is not APF-authorized in the STEPLIB, the whole library concatenation loses its APF authorization.

The APF lists are in the SYS1.PARMLIB member PROGxx or IEAAPFaa. The lists contain the names of APF authorized z/OS libraries. The order of the entries in the lists is not significant. See the *MVS Initialization and Tuning Reference* manual for information about APF lists.

For more information about tuning your system, see SupportPac MP16

If you use PROGxx members with dynamic format, you need only issue the z/OS command `SETPROG APF,ADD,DSNAME=hlq.SCSQ` *XXXX*`,VOLUME=` *YYYYYY* for the changes to take effect: Where *XXXX* varies by the library name and where *YYYYY* is the volume. Otherwise, if you use static format or IEAAPFaa members, you must perform an IPL on your system.

Note that you must use the actual name of the library in the APF list. If you attempt to use the data set alias of the library, authorization fails.

Related concepts

"Task 3: Update the z/OS link list and LPA" on page 531
Update the LPA libraries with the new version of early code libraries. Other code can go in the link list or the LPA.

"Preparing to customize your IBM MQ for z/OS queue managers" on page 522
Use this topic when customizing your queue managers with details of installable features, national language features, and information about testing, and setting up security.

Task 3: Update the z/OS link list and LPA

Update the LPA libraries with the new version of early code libraries. Other code can go in the link list or the LPA.

- You need to perform this task once for each z/OS system where you want to run IBM MQ.
- If you are using queue-sharing groups, you must ensure that the settings for IBM MQ are identical on each z/OS system in the sysplex.
- You might need to perform this task when migrating from a previous version. For details, see the Program Directory for WebSphere MQ for z/OS.

Note: The data set for LPA is version specific. If you are using an existing LPA in the system, contact your system administrator to decide which LPA to use.

Early code

Some IBM MQ load modules need to be added to MVS for IBM MQ to act as a subsystem. These modules are known as the Early code, and they can be executed even if a queue manager is not active. For example, when an operator command is issued on the console with an IBM MQ command prefix, this Early code will get control and check if it needs to start a queue manager, or to pass the request to a running queue manager. This code is loaded into the Link Pack Area (LPA). There is one set of Early modules, which are used for all queue managers, and these need to be at the highest level of IBM MQ. Early code from a higher version of IBM MQ will work with a queue manager with a lower version of IBM MQ, but not the opposite.

IBM MQ

The early code comprises the following load modules:

- CSQ3INI and CSQ3EPX in the library thqual.SCSQLINK
- CSQ3ECMX in the library thqual.SCSQSNL *x*, where *x* is your language letter.

IBM MQ includes a user modification that moves the contents of the thqual.SCSQSNL *x* library into the thqual.SCSQLINK and informs SMP/E. This user modification is called CSQ8UERL and is described in the Program Directory for WebSphere MQ for z/OS.

When you have updated the early code in the LPA libraries, it is available from the next z/OS IPL (with the CLPA option) to all queue manager subsystems added during IPL from definitions in IEFSSNss members in SYS1.PARMLIB.

You can make it available immediately without an IPL for any new queue manager subsystem added later (as described in "Task 5: Define the IBM MQ subsystem to z/OS" on page 533) by adding it to the LPA as follows:

- If you did not use CSQ8UERL, issue these z/OS commands:

```
SETPROG LPA,ADD,MODNAME=(CSQ3INI,CSQ3EPX),DSNAME=thqual.SCSQLINK
SETPROG LPA,ADD,MODNAME=(CSQ3ECMX),DSNAME=thqual.SCSQSNL x
```

- If you did use CSQ8UERL, you can load the early code into the LPA using the following z/OS command:

```
SETPROG LPA,ADD,MASK=*,DSNAME=thqual.SCSQLINK
```

- If you are using IBM MQ Advanced Message Security you must also issue the following z/OS command to include an additional module in the LPA:

```
SETPROG LPA,ADD,MODNAME=(CSQ0DRTM),DSNAME=thqual.SCSQLINK
```

If you have applied maintenance, or you intend to restart a queue manager with a later version or release of IBM MQ, the early code can be made available to queue manager subsystems that are already defined. To make it available, use the following steps:

1. Add it to the LPA using z/OS SETPROG commands as described previously in this topic.
2. Stop the queue manager, using the IBM MQ command STOP QMGR.
3. Ensure that the qmgr.REFRESH.QMGR security profile is set up. See MQSC commands, profiles, and their access levels.
4. Refresh the early code for the queue manager using the IBM MQ command REFRESH QMGR TYPE(EARLY).
5. Restart the queue manager, using the IBM MQ command START QMGR.

The IBM MQ commands STOP QMGR, REFRESH QMGR, and START QMGR are described in MQSC commands.

Other code

All the IBM MQ supplied load modules in the following libraries are reentrant and can be placed in the LPA:

- SCSQAUTH
- SCSQANL *x*, where *x* is your language letter
- SCSQMVR1

Important: However, if you place the libraries in the LPA, whenever you apply maintenance, you have to copy any changed modules manually into the LPA. Therefore, it is preferable to put the IBM MQ load libraries in the link list, which can be updated after maintenance by issuing the z/OS command REFRESH LLA.

This is particularly recommended for SCSQAUTH so that you do not have to include it in several STEPLIBs. Only one language library, SCSQANL *x* should be placed in the LPA or link list. The link list libraries are specified in an LNKLSTkk member of SYS1.PARMLIB.

The distributed queuing facility and CICS bridge (but not the queue manager itself) need access to the Language Environment (LE) runtime library SCEERUN. If you use either of these facilities, you need to include SCEERUN in the link list.

Related concepts

"Task 4: Update the z/OS program properties table" on page 533
Some additional PPT entries are needed for the IBM MQ queue manager.

Task 4: Update the z/OS program properties table

Some additional PPT entries are needed for the IBM MQ queue manager.

- *You must perform this task once for each z/OS system where you want to run IBM MQ.*
- *If you are using queue-sharing groups, you must ensure that the settings for IBM MQ are identical on each z/OS system in the sysplex.*
- *You do not need to perform this task when migrating from a previous version.*
- *You do need to perform the CSQ0DSRV part of this task when you require IBM MQ Advanced Message Security.*

A sample containing all the required PPT entries is provided in thlqual.SCSQPROC(CSQ4SCHD). Ensure that the required entries are added to the PPT, which you can find in SYS1.PARMLIB(SCHEDxx).

In z/OS 1.12 and later versions, CSQYASCP is already defined to the operating system with the attributes detailed and no longer needs to be included in a SCHEDxx member of PARMLIB.

The IBM MQ queue manager controls swapping itself. However, if you have a heavily-loaded IBM MQ network and response time is critical, it might be advantageous to make the IBM MQ channel initiator nonswappable, by adding the CSQXJST PPT entry, at the risk of affecting the performance of the rest of your z/OS system.

If you require IBM MQ Advanced Message Security, add the CSQ0DSRV PPT entry.

Issue the z/OS command SET SCH= for these changes to take effect.

Related concepts

"Task 5: Define the IBM MQ subsystem to z/OS" on page 533
Update the subsystem name table and decide on a convention for command prefix strings.

Task 5: Define the IBM MQ subsystem to z/OS

Update the subsystem name table and decide on a convention for command prefix strings.

Repeat this task for each IBM MQ queue manager. You do not need to perform this task when migrating from a previous version.

Related concepts

"Task 6: Create procedures for the IBM MQ queue manager" on page 537
Each IBM MQ subsystem needs a cataloged procedure to start the queue manager. You can create your own or use the IBM-supplied procedure library.

Updating the subsystem name table

When defining the IBM MQ subsystem you must add an entry to the subsystem name table.

The subsystem name table of z/OS, which is taken initially from the SYS1.PARMLIB member IEFSSNss, contains the definitions of formally defined z/OS subsystems. To define each IBM MQ subsystem, you must add an entry to this table, either by changing the IEFSSNss member of SYS1.PARMLIB, or, preferably, by using the z/OS command SETSSI.

IBM MQ subsystem initialization supports parallel processing, so IBM MQ subsystem definition statements can be added both above and below the BEGINPARALLEL keyword in the IEFSSNss table available at z/OS V1.12 and later.

If you use the SETSSI command, the change takes effect immediately, and there is no need to perform an IPL of your system. Ensure you update SYS1.PARMLIB as well, as described in "Task 12: Update SYS1.PARMLIB members" on page 543 so that the changes remain in effect after subsequent IPLs.

The SETSSI command to dynamically define an IBM MQ subsystem is:

```
SETSSI ADD,S=ssid,I=CSQ3INI,P='CSQ3EPX,cpf,scope'
```

The corresponding information in IEFSSNss can be specified in one of two ways:

- The keyword parameter form of the IBM MQ subsystem definition in IEFSSNss. This is the recommended method.

  ```
  SUBSYS SUBNAME(ssid) INITRTN(CSQ3INI) INITPARM('CSQ3EPX,cpf,scope')
  ```

- The positional parameter form of the IBM MQ subsystem definition.

  ```
  ssid,CSQ3INI,'CSQ3EPX,cpf,scope'
  ```

Do not mix the two forms in one IEFSSNss member. If different forms are required, use a separate IEFSSNss member for each type, adding the SSN operand of the new member to the IEASYSpp SYS1.PARMLIB member. To specify more than one SSN, use SSN=(aa,bb,...) in IEASYSpp.

In the examples,

ssid
: The subsystem identifier. It can be up to four characters long. All characters must be alphanumeric (uppercase A through Z, 0 through 9), it must start with an alphabetic character. The queue manager will have the same name as the subsystem, therefore you can use only characters that are allowed for both z/OS subsystem names and IBM MQ object names.

cpf
: The command prefix string (see "Defining command prefix strings (CPFs)" on page 535 for information about CPFs).

scope
: The system scope, used if you are running in a z/OS sysplex (see "CPFs in a sysplex environment" on page 536 for information about system scope).

Figure 90 on page 534 shows several examples of IEFSSNss statements.

```
CSQ1,CSQ3INI,'CSQ3EPX,+mqs1cpf,S'
CSQ2,CSQ3INI,'CSQ3EPX,+mqs2cpf,S'
CSQ3,CSQ3INI,'CSQ3EPX,++,S'
```

Figure 90. Sample IEFSSNss statements for defining subsystems

Note: When you have created objects in a subsystem, you cannot change the subsystem name or use the page sets from one subsystem in another subsystem. To do either of these, you must unload all the objects and messages from one subsystem and reload them into another.

Table 38 on page 535 gives a number of examples showing the associations of subsystem names and command prefix strings (CPFs), as defined by the statements in Figure 90 on page 534.

Table 38. Subsystem name to CPF associations	
IBM MQ subsystem name	**CPF**
CSQ1	+mqs1cpf
CSQ2	+mqs2cpf
CSQ3	++

Note: The ACTIVATE and DEACTIVATE functions of the z/OS command SETSSI are not supported by IBM MQ.

To check the status of the changes, issue the following command in SDSF: `/D SSI,L`. You will see the new subsystems created with ACTIVE status.

Defining command prefix strings (CPFs)

Each subsystem instance of IBM MQ can have a command prefix string to identify that subsystem.

Adopt a system-wide convention for your CPFs for all subsystems to avoid conflicts. Adhere to the following guidelines:

- Define a CPF as string of up to eight characters.
- Do not use a CPF that is already in use by any other subsystem, and avoid using the JES backspace character defined on your system as the first character of your string.
- Define your CPF using characters from the set of valid characters listed in Table 40 on page 536.
- Do not use a CPF that is an abbreviation for an already defined process or that might be confused with command syntax. For example, a CPF such as 'D' conflicts with z/OS commands such as DISPLAY. To avoid this happening, use one of the special characters (shown in Table 40 on page 536) as the first or only character in your CPF string.
- Do not define a CPF that is either a subset or a superset of an existing CPF. For an example, see Table 39 on page 535.

Table 39. Example of CPF subset and superset rules		
Subsystem name	**CPF defined**	**Commands routed to**
MQA	!A	MQA
MQB	!B	MQB
MQC1	!C1	MQC1
MQC2	!C2	MQC2
MQB1	!B1	MQB

Commands intended for subsystem MQB1 (using CPF !B1) are routed to subsystem MQB because the CPF for this subsystem is !B, a subset of !B1. For example, if you entered the command:

```
!B1  START  QMGR
```

subsystem MQB receives the command:

```
1 START QMGR
```

(which, in this case, it cannot deal with).

You can see which prefixes exist by issuing the z/OS command DISPLAY OPDATA.

If you are running in a sysplex, z/OS diagnoses any conflicts of this type at the time of CPF registration (see “CPFs in a sysplex environment” on page 536 for information about CPF registration).

Table 40 on page 536 shows the characters that you can use when defining your CPF strings:

Table 40. Valid character set for CPF strings

Character set	Contents
Alphabetic	Uppercase A through Z, lowercase a through z
Numeric	0 through 9
National (see note)	@ $ # (Characters that can be represented as hexadecimal values)
Special	. □ ( ) * & + - = ¢ < | ! ; % _ ? : >

Note:

The system recognizes the following hexadecimal representations of the national characters: @ as X'7C', $ as X'5B', and # as X'7B'. In countries other than the U.S., the U.S. national characters represented on terminal keyboards might generate a different hexadecimal representation and cause an error. For example, in some countries the $ character might generate an X'4A'.

The semicolon (;) is valid as a CPF but on most systems, this character is the command delimiter.

CPFs in a sysplex environment

Use this topic to understand how to use CPFs within the scope of a sysplex.

If used in a sysplex environment, IBM MQ registers your CPFs to enable you to enter a command from any console in the sysplex and route that command to the appropriate system for execution. The command responses are returned to the originating console.

Defining the scope for sysplex operation

Scope is used to determine the type of CPF registration performed by the IBM MQ subsystem when you are running IBM MQ in a sysplex environment.

Possible values for scope are as follows:

M

System scope.

The CPF is registered with z/OS at system IPL time by IBM MQ and remains registered for the entire time that the z/OS system is active.

IBM MQ commands must be entered at a console connected to the z/OS image running the target subsystem, or you must use ROUTE commands to direct the command to that image.

Use this option if you are not running in a sysplex.

S

Sysplex started scope.

The CPF is registered with z/OS when the IBM MQ subsystem is started, and remains active until the IBM MQ subsystem terminates.

You must use ROUTE commands to direct the original START QMGR command to the target system, but all further IBM MQ commands can be entered at any console connected to the sysplex, and are routed to the target system automatically.

After IBM MQ termination, you must use the ROUTE commands to direct subsequent START commands to the target IBM MQ subsystem.

X

Sysplex IPL scope.

The CPF is registered with z/OS at system IPL time by IBM MQ and remains registered for the entire time that the z/OS system is active.

IBM MQ commands can be entered at any console connected to the sysplex, and are routed to the image that is executing the target system automatically.

An IBM MQ subsystem with a CPF with scope of S can be defined on one or more z/OS images within a sysplex, so these images can share a single subsystem name table. However, you must ensure that the initial START command is issued on (or routed to) the z/OS image on which you want the IBM MQ subsystem to run. If you use this option, you can stop the IBM MQ subsystem and restart it on a different z/OS image within the sysplex without having to change the subsystem name table or perform an IPL of a z/OS system.

An IBM MQ subsystem with a CPF with scope of X can only be defined on one z/OS image within a sysplex. If you use this option, you must define a unique subsystem name table for each z/OS image requiring IBM MQ subsystems with CPFs of scope X.

If you want to use the z/OS automatic restart manager (ARM) to restart queue managers in different z/OS images automatically, every queue manager must be defined in each z/OS image on which that queue manager might be restarted. Every queue manager must be defined with a sysplex-wide, unique 4-character subsystem name with a CPF scope of S.

Task 6: Create procedures for the IBM MQ queue manager

Each IBM MQ subsystem needs a cataloged procedure to start the queue manager. You can create your own or use the IBM-supplied procedure library.

- *Repeat this task for each IBM MQ queue manager.*
- *You might need to modify the catalogued procedure when migrating from a previous version.*

For each IBM MQ subsystem defined in the subsystem name table, create a cataloged procedure in a procedure library for starting the queue manager. The IBM-supplied procedure library is called SYS1.PROCLIB, but your installation might use its own naming convention.

The name of the queue manager started task procedure is formed by concatenating the subsystem name with the characters MSTR. For example, subsystem CSQ1 has the procedure name CSQ1MSTR. You need one procedure for each subsystem you define.

Many examples and instructions in this product documentation assume that you have a subsystem called CSQ1 You might find these examples easier to use if a subsystem called CSQ1 is created initially for installation verification and testing purposes.

Two sample started task procedures are provided in thlqual.SCSQPROC. Member CSQ4MSTR uses one page set for each class of message, member CSQ4MSRR uses multiple page sets for the major classes of message. Copy one of these procedures to member xxxxMSTR (where xxxx is the name of your IBM MQ subsystem) of your SYS1.PROCLIB or, if you are not using SYS1.PROCLIB, your procedure library. Copy the sample procedure to a member in your procedure library for each IBM MQ subsystem that you define.

When you have copied the members, you can tailor them to the requirements of each subsystem, using the instructions in the member. For information about specifying region sizes below the 16 MB line, above the 16 MB line, and above the 2 GB bar, see Suggested region sizes. You can also use symbolic parameters in the JCL to allow the procedure to be modified when it is started. If you have several IBM MQ subsystems, you might find it advantageous to use JCL include groups for the common parts of the procedure, to simplify future maintenance.

If you are using queue-sharing groups, the STEPLIB concatenation must include the Db2 runtime target library SDSNLOAD, and it must be APF-authorized. This library is only required in the STEPLIB concatenation if it is not accessible through the link list or LPA.

If you are using IBM MQ Advanced Message Security the STEPLIB concatenation must include *thlqual*.SDRQAUTH and it must be APF authorized.

You can add the exit library (CSQXLIB) to this procedure later if you want to use queue manager exits. You need access to the Language Environment (LE) runtime library SCEERUN to do this; if it is not in your link list (SYS1.PARMLIB(LNKLSTkk)), concatenate it in the STEPLIB DD statement. You also must stop and restart your queue manager.

Note: You can make a note of the names of your bootstrap data set (BSDS), logs, and page sets for use in JCL and then define these sets at a later step in the process.

Related concepts

"Task 7: Create procedures for the channel initiator" on page 538
For each IBM MQ subsystem, tailor a copy of CSQ4CHIN. Depending on what other products you are using, you might need to allow access to other data sets.

Task 7: Create procedures for the channel initiator

For each IBM MQ subsystem, tailor a copy of CSQ4CHIN. Depending on what other products you are using, you might need to allow access to other data sets.

- *Repeat this task for each IBM MQ queue manager.*
- *You might need to perform this task when migrating from a previous version.*

You need to create a channel-initiator started task procedure for each IBM MQ subsystem that is going to use distributed queuing.

To do this:

1. Copy the sample started task procedure thlqual.SCSQPROC(CSQ4CHIN) to your procedure library. Name the procedure *xxxx* CHIN, where *xxxx* is the name of your IBM MQ subsystem (for example, CSQ1CHIN would be the channel initiator started task procedure for queue manager CSQ1).
2. Make a copy for each IBM MQ subsystem that you are going to use.
3. Tailor the procedures to your requirements using the instructions in the sample procedure CSQ4CHIN. You can also use symbolic parameters in the JCL to allow the procedure to be modified when it is started. This is described with the start options in Administering IBM MQ for z/OS.

   Concatenate the distributed queuing library thlqual.SCSQMVR1.

   Access to the LE runtime library SCEERUN is required; if it is not in your link list (SYS1.PARMLIB(LNKLSTkk)), concatenate it in the STEPLIB DD statement.
4. Authorize the procedures to run under your external security manager.

The channel initiator is a long running address space. To prevent its termination after a restricted amount of CPU has been consumed, confirm that either:

- The default for started tasks in your z/OS system is unlimited CPU; a JES2 configuration statement for JOBCLASS(STC) with TIME=(1440,00) achieves this, or
- Explicitly add a TIME=1440, or TIME=NOLIMIT, parameter to the EXEC statement for CSQXJST.

You can add the exit library (CSQXLIB) to this procedure later if you want to use channel exits. You need to stop and restart your channel initiator to do this.

If you are using SSL, access to the system Secure Sockets Layer (SSL) runtime library is required. This library is called SIEALNKE. The library must be APF authorized.

If you are using TCP/IP, the channel initiator address space must be able to access the TCPIP.DATA data set that contains TCP/IP system parameters. The ways that the data set has to be set up depends on which TCP/IP product and interface you are using. They include:

- Environment variable, RESOLVER_CONFIG
- HFS file, /etc/resolv.conf
- //SYSTCPD DD statement
- //SYSTCPDD DD statement
- *jobname/userid*.TCPIP.DATA
- SYS1.TCPPARMS(TCPDATA)

- *zapname*.TCPIP.DATA

Some of these affect your started-task procedure JCL. For more information, see z/OS Communications Server: IP Configuration Guide.

Related concepts

"Task 8: Define the IBM MQ subsystem to a z/OS WLM service class" on page 539
To give IBM MQ appropriate performance priority in the z/OS system, you must assign the queue manager and channel initiator address spaces to an appropriate z/OS workload management (WLM) service class. If you do not do this explicitly, inappropriate defaults might apply.

Task 8: Define the IBM MQ subsystem to a z/OS WLM service class

To give IBM MQ appropriate performance priority in the z/OS system, you must assign the queue manager and channel initiator address spaces to an appropriate z/OS workload management (WLM) service class. If you do not do this explicitly, inappropriate defaults might apply.

- *Repeat this task for each IBM MQ queue manager.*
- *You do not need to perform this task when migrating from a previous version.*

Use the ISPF dialog supplied with WLM to perform the following tasks:

- Extract the z/OS WLM policy definition from the WLM couple data set.
- Update this policy definition by adding queue manager and channel initiator started task procedure names to the chosen service class
- Install the changed policy on the WLM couple data set

Then activate this policy using the z/OS command

```
V WLM,POLICY=policyname,REFRESH
```

See for more information on setting performance options.

Related concepts

"Task 9: Set up the Db2 environment" on page 539
If you are using queue-sharing groups you must create the required Db2 objects by customizing and running a number of sample jobs.

Task 9: Set up the Db2 environment

If you are using queue-sharing groups you must create the required Db2 objects by customizing and running a number of sample jobs.

For more information about selecting which offload storage environment to use, see Deciding your offload storage environment.

If you choose Db2 as the offload storage environment, see "Set up the Db2 environment" on page 539. If you choose SMDS as the offload storage environment, see Set up the SMDS environment. If you are choosing the SMDS offload storage environment, you are still required to set up the Db2 environment for shared queues.

Set up the Db2 environment

You must create and bind the required Db2 objects by customizing and running a number of sample jobs.

- Repeat this task for each Db2 data-sharing group.
- You might need to perform this task when migrating from a previous version.
- Omit this task if you are not using queue-sharing groups.

  If you later want to use queue-sharing groups, perform this task at that time.

IBM MQ provides two equivalent sets of jobs. Those with the CSQ45 prefix are for compatibility with earlier versions of IBM MQ and for use with Db2 version 11 and earlier. If you are setting up a new data-sharing group with Db2 V12 or later, you are encouraged to use the jobs with CSQ4X prefix, as these jobs exploit more recent Db2 capabilities for dynamic sizing and Universal Table Spaces.

You must establish an environment in which IBM MQ can access and execute the Db2 plans that are used for queue-sharing groups.

The following steps must be performed for each new Db2 data-sharing group. All the sample JCL is in thlqual.SCSQPROC.

1. Customize and execute sample JCL CSQ45CSG (or CSQ4XCSG) to create the storage group that is to be used for the IBM MQ database, table spaces, and tables.
2. Customize and execute sample JCL CSQ45CDB (or CSQ4XCDB) to create the database to be used by all queue managers that are connecting to this Db2 data-sharing group.
3. Customize and execute sample JCL CSQ45CTS (or CSQ4XCTS) to create the table spaces that contain the queue manager and channel initiator tables used for queue-sharing groups (to be created in step 1).
4. Customize and execute sample JCL CSQ45CTB (or CSQ4XCTB) to create the 12 Db2 tables and associated indexes. Do not change any of the row names or attributes.
5. Customize and execute sample JCL CSQ45BPL (or CSQ4XBPL)to bind the Db2 plans for the queue manager, utilities, and channel initiator.
6. Customize and execute sample JCL CSQ45GEX (or CSQ4XGEX)to grant execute authority to the plans for the user IDs that are used by the queue manager, utilities, and channel initiator. The user IDs for the queue manager and channel initiator are the user IDs under which their started task procedures run. The user IDs for the utilities are the user IDs under which the batch jobs can be submitted. The names of the appropriate plans are:

User	Plans
Queue manager	CSQ5A 800, CSQ5C 800, CSQ5D 800, CSQ5K 800, CSQ5L 800, CSQ5M 800, CSQ5P 800, CSQ5R 800, CSQ5S 800, CSQ5T 800, CSQ5U 800, CSQ5W 800
SDEFS function of the CSQUTIL batch utility	CSQ52 800
CSQ5PQSG and CSQJUCNV batch utilities	CSQ5B 800
CSQUZAP service utility	CSQ5Z 800

In the event of a failure during Db2 setup, the following jobs can be customized and executed:

- CSQ45DTB to drop the tables and indexes.
- CSQ45DTS to drop the table spaces.
- CSQ45DDB to drop the database.
- CSQ45DSG to drop the storage group.

Note: If these jobs fail because of a Db2 locking problem it is probably due to contention for a Db2 resource, especially if the system is being heavily used. Resubmit the jobs later. It is preferable to run these jobs when the system is lightly used or quiesced.

See *Db2 10 for z/OS: Db2 Administration* for more information about setting up Db2.

See Planning on z/OS for information about Db2 table sizes.

Set up the SMDS environment

You should use SMDS to offload large messages. You can offload all messages to SMDS to give you more capacity in your structure.

You can use Storage Class Memory(SCM); see Use of storage class memory with shared queues.

- Estimate structure and data set space requirements. See Shared message data set capacity considerations.
- Allocate and preformat data sets. See Creating a shared message data set.
- Use the following MQSC command to display the **CFLEVEL** and **OFFLOAD** status.

  ```
  DISPLAY CFSTRUCT(*) CFLEVEL OFFLOAD
  ```

  For more information about the **DISPLAY CFSTRUCT** command, see DISPLAY CFSTRUCT.
- Ensure that the coupling facility structure is defined with **CFLEVEL(5)** and **OFFLOAD(SMDS)** by using the following MQSC commands:

  ```
  ALTER CFSTRUCT(APP1) CFLEVEL(5)
  ALTER CFSTRUCT(APP1) OFFLOAD(SMDS)
  ```

  For more information about the **ALTER CFSTRUCT** command, see ALTER CFSTRUCT.

Related concepts

"Task 10: Set up the coupling facility" on page 541
If you are using queue-sharing groups, define the coupling facility structures used by the queue managers in the queue-sharing group in the coupling facility Resource Management (CFRM) policy data set, using IXCMIAPU.

Task 10: Set up the coupling facility

If you are using queue-sharing groups, define the coupling facility structures used by the queue managers in the queue-sharing group in the coupling facility Resource Management (CFRM) policy data set, using IXCMIAPU.

- *Repeat this task for each queue-sharing group.*
- *You might need to perform this task when migrating from a previous version.*
- *Omit this task if you are not using queue-sharing groups.*

  If you later want to use queue-sharing groups, perform this task at that time.

All the structures for the queue-sharing group start with the name of the queue-sharing group. Define the following structures:

- An administrative structure called *qsg-name* CSQ_ADMIN. This structure is used by IBM MQ itself and does not contain any user data.
- A system application structure called *qsg-name* CSQSYSAPPL. This structure is used by IBM MQ system queues to store state information.
- One or more structures used to hold messages for shared queues. These can have any name you choose up to 16 characters long.
  - The first four characters must be the queue-sharing group name. (If the queue-sharing group name is less than four characters long, it must be padded to four characters with @ symbols.)
  - The fifth character must be alphabetic and subsequent characters can be alphabetic or numeric. This part of the name (without the queue-sharing group name) is what you specify for the CFSTRUCT name when you define a shared queue, or a CF structure object.

  You can use only alphabetic and numeric characters in the names of structures used to hold messages for shared queues, you cannot use any other characters (for example, the _ character, which is used in the name of the administrative structure).

Sample control statements for IXCMIAPU are in data set thlqual.SCSQPROC(CSQ4CFRM). Customize these and add them to your IXCMIAPU job for the coupling facility and run it.

When you have defined your structures successfully, activate the CFRM policy that is being used. To do this, issue the following z/OS command:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME= policy-name
```

See the Defining coupling facility resources for information about planning CF structures and their sizes.

Related concepts

"Task 11: Implement your ESM security controls" on page 542
Implement security controls for queue-sharing groups, the channel initiator, and all queue managers accessing the coupling facility list structures.

Task 11: Implement your ESM security controls

Implement security controls for queue-sharing groups, the channel initiator, and all queue managers accessing the coupling facility list structures.

- *Repeat this task for each IBM MQ queue manager or queue-sharing group.*
- *You might need to perform this task when migrating from a previous version.*

If you use RACF® as your external security manager, see Setting up security on z/OS , which describes how to implement these security controls.

If you are using queue-sharing groups, ensure that the user IDs associated with the queue manager, channel initiator, and the utilities (as specified in task 9, step "6" on page 540) have authority to establish an RRSAF connection to each Db2 subsystem with which you want to establish a connection. The RACF profile to which the user ID requires READ access is *DB2ssid*.RRSAF in the DSNR resource class.

If you are using the channel initiator, you must also do the following:

- If your subsystem has connection security active, define a connection security profile ssid.CHIN to your external security manager (see Connection security profiles for the channel initiator for information about this).
- If you are using the Secure Sockets Layer (SSL) or a sockets interface, ensure that the user ID under whose authority the channel initiator is running is configured to use UNIX System Services, as described in the *OS/390® UNIX System Services Planning* documentation.
- If you are using SSL, ensure that the user ID under whose authority the channel initiator is running is configured to access the key ring specified in the SSLKEYR parameter of the ALTER QMGR command.

Those queue managers that will access the coupling facility list structures require the appropriate security access. The RACF class is FACILITY. The queue manager user ID requires ALTER access to the IXLSTR. `structure-name` profile.

Before you start the queue manager, set up IBM MQ data set and system security by:

- Authorizing the queue manager started task procedure to run under your external security manager.
- Authorizing access to the queue manager data sets.

For details about how to do this, see Security installation tasks for z/OS(r).

If you are using RACF, provided you use the RACF STARTED class, you do not need to perform an IPL of your system (see RACF authorization of started-task procedures).

Related concepts

"Task 12: Update SYS1.PARMLIB members" on page 543

To ensure that your changes remain in effect after an IPL, you must update some members of SYS1.PARMLIB

Task 12: Update SYS1.PARMLIB members

To ensure that your changes remain in effect after an IPL, you must update some members of SYS1.PARMLIB

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *If you are using queue-sharing groups, you must ensure that the settings for IBM MQ are identical on each z/OS system in the sysplex.*
- *You might need to perform this task when migrating from a previous version.*

Update SYS1.PARMLIB members as follows:

1. Update member IEFSSNss as described in "Task 5: Define the IBM MQ subsystem to z/OS" on page 533.
2. Change IEASYSpp so that the following members are used when an IPL is performed:
   - the PROGxx or IEAAPFaa members used in "Task 2: APF authorize the IBM MQ load libraries" on page 530
   - the LNKLSTkk and LPALSTmm members used in "Task 3: Update the z/OS link list and LPA" on page 531
   - the SCHEDxx member used in "Task 4: Update the z/OS program properties table" on page 533
   - the IEFSSNss member used in "Task 5: Define the IBM MQ subsystem to z/OS" on page 533

Related concepts

"Task 13: Customize the initialization input data sets" on page 543
Make working copies of the sample initialization input data sets and tailor them to suit your system requirements.

Task 13: Customize the initialization input data sets

Make working copies of the sample initialization input data sets and tailor them to suit your system requirements.

- *Repeat this task for each IBM MQ queue manager.*
- *You need to perform this task when migrating from a previous version.*

Each IBM MQ queue manager gets its initial definitions from a series of commands contained in the IBM MQ *initialization input data sets*. These data sets are referenced by the DDnames CSQINP1, CSQINP2 and CSQINPT defined in the queue manager started task procedure.

Responses to these commands are written to the initialization output data sets referenced by the DDnames CSQOUT1, CSQOUT2 and CSQOUTT.

To preserve the originals, make working copies of each sample. Then you can tailor the commands in these working copies to suit your system requirements.

If you use more than one IBM MQ subsystem, if you include the subsystem name in the high-level qualifier of the initialization input data set name, you can identify the IBM MQ subsystem associated with each data set more easily.

Refer to the following topics for further information about the samples:

- Initialization data set formats
- Using the CSQINP1 sample
- Using the CSQINP2 samples
- Using the CSQINPX sample

- Using the CSQINPT sample

Initialization data set formats

The initialization input data sets can be partitioned data set (PDS) members or sequential data sets. They can be a concatenated series of data sets. Define them with a record length of 80 bytes, where:

- Only columns 1 through 72 are significant. Columns 73 through 80 are ignored.
- Records with an asterisk (*) in column 1 are interpreted as comments and are ignored.
- Blank records are ignored.
- Each command must start on a new record.
- A trailing - means continue from column 1 of the next record.
- A trailing + means continue from the first non-blank column of the next record.
- The maximum number of characters permitted in a command is 32 762.

The initialization output data sets are sequential data sets, with a record length of 125, a record format of VBA, and a block size of 629.

Using the CSQINP1 sample

Data set thlqual.SCSQPROC holds two members which contain definitions of buffer pools, page set to buffer pool associations, and an ALTER SECURITY command.

Member CSQ4INP1 uses one page set for each class of message. Member CSQ4INPR uses multiple page sets for the major classes of message.

Include the appropriate sample in the CSQINP1 concatenation of your queue manager started task procedure.

Notes:

1. IBM MQ supports up to 16 buffer pools (zero through 15). If OPMODE is set to OPMODE=(NEWFUNC, 800), 100 buffer pools are supported in the range zero through 99. The DEFINE BUFFPOOL command can only be issued from a CSQINP1 initialization data set. The definitions in the sample specify four buffer pools.
2. Each page set used by the queue manager must be defined in the CSQINP1 initialization data set by using the DEFINE PSID command. The page set definition associates a buffer pool ID with a page set. If no buffer pool is specified, buffer pool zero is used by default.

   Page set zero (00) must be defined. It contains all the object definitions. You can define up to 100 page sets for each queue manager.
3. The ALTER SECURITY command can be used to alter the security attributes TIMEOUT and INTERVAL. In CSQ4INP1, the default values are defined as 54 for TIMEOUT and 12 for INTERVAL.

See the Planning on z/OS for information about organizing buffer pools and page sets.

If you change the buffer pool and page set definitions dynamically while the queue manager is running, you should also update the CSQINP1 definitions. The changes are only retained for a cold start of IBM MQ, unless the buffer pool definition includes the REPLACE attribute.

Using the CSQINP2 samples

This table lists the members of thlqual.SCSQPROC that can be included in the CSQINP2 concatenation of your queue manager started task procedure, with a description of their function. The naming convention is CSQ4INS*. CSQ4INY* will need to be modified for YOUR configuration. You should avoid changing CSQINS* members because you will need to reapply any changes when you migrate to the next release. Instead, you can put DEFINE or ALTER commands in CSQ4INY* members.

Table 41. Members of thlqual.SCSQPROC	
Member name	**Description**
CSQ4INSG	System object definitions.
CSQ4INSA	System object and default rules for channel authentication.
CSQ4INSX	System object definitions.
CSQ4INSS	Customize and include this member if you are using queue-sharing groups.
CSQ4INSJ	Customize and include this member if you are using publish/subscribe using JMS.
CSQ4INSM	System object definitions for advanced message security.
CSQ4INSR	Customize and include this member if you are using WebSphere Application Server, or the queued publish/subscribe interface supported by the queued publish/ subscribe daemon in IBM MQ V7 or later.
CSQ4DISP	CSQINP2 sample for displaying object definitions.
CSQ4INYC	Clustering definitions.
CSQ4INYD	Distributed queuing definitions.
CSQ4INYG	General definitions.
CSQ4INYR	Storage class definitions, using multiple page sets for the major classes of message.
CSQ4INYS	Storage class definitions, using one page set for each class of message.

You need to define objects once only, not each time that you start a queue manager, so it is not necessary to include these definitions in CSQINP2 every time. If you do include them every time, you are attempting to define objects that already exist, and you will get messages similar to the following:

```
CSQM095I +CSQ1 CSQMAQLC QLOCAL(SYSTEM.DEFAULT.LOCAL.QUEUE) ALREADY EXISTS
CSQM090E +CSQ1 CSQMAQLC FAILURE REASON CODE X'00D44003'
CSQ9023E +CSQ1 CSQMAQLC ' DEFINE QLOCAL' ABNORMAL COMPLETION
```

The objects are not damaged by this failure. If you want to leave the SYSTEM definitions data set in the CSQINP2 concatenation, you can avoid the failure messages by specifying the REPLACE attribute against each object.

Using the CSQINPX sample

Sample thlqual.SCSQPROC(CSQ4INPX) contains a set of commands that you might want to execute each time the channel initiator starts. These are typically channel-related commands such as START LISTENER, which are required every time the channel initiator starts, rather than whenever the queue manager starts, and which are not allowed in the input data sets CSQINP1 or CSQINP2. You must customize this sample before use; you can then include it in the CSQINPX data set for the channel initiator.

The IBM MQ 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. The output is like that produced by the COMMAND function of the IBM MQ utility program (CSQUTIL). See The CSQUTIL utility for more details.

You can specify any of the IBM MQ commands that can be issued from CSQUTIL, not only the channel commands. You can enter commands from other sources while CSQINPX is being processed. All commands are issued in sequence, regardless of the success of the previous command.

To specify a command response time, you can use the pseudo-command COMMAND as the first command in the data set. This takes a single optional keyword RESPTIME(*nnn*), where *nnn* is the time, in seconds, to wait for a response to each command. This is in the range 5 through 999; the default is 30.

If IBM MQ detects that the responses to four commands have taken too long, processing of CSQINPX is stopped and no further commands are issued. The channel initiator is not stopped, but message CSQU052E is written to the CSQOUTX data set, and message CSQU013E is sent to the console.

When IBM MQ has completed processing of CSQINPX successfully, message CSQU012I is sent to the console.

Using the CSQINPT sample

This table lists the members of thlqual.SCSQPROC that can be included in the CSQINPT concatenation of your queue manager started task procedure, with a description of their function.

Table 42. Members of thlqual.SCSQPROC

Member name	Description
CSQ4INST	System default subscription definition.
CSQ4INYT	Publish/Subscribe definitions.

The IBM MQ commands contained in the data set are executed when publish/subscribe initialization completes, and output is written to the data set specified by the CSQOUTT DD statement. The output is like that produced by the COMMAND function of the IBM MQ utility program (CSQUTIL). See The CSQUTIL utility for more details.

Related concepts

"Task 14: Create the bootstrap and log data sets" on page 546
Use the supplied program CSQJU003 to prepare the bootstrap data sets (BSDSs) and log data sets.

Task 14: Create the bootstrap and log data sets

Use the supplied program CSQJU003 to prepare the bootstrap data sets (BSDSs) and log data sets.

- *Repeat this task for each IBM MQ queue manager.*
- *You do not need to perform this task when migrating from a previous version.*

The sample JCL and Access Method Services (AMS) control statements to run CSQJU003 to create a single or dual logging environment are held in thlqual.SCSQPROC(CSQ4BSDS). Customize and run this job to create your BSDSs and logs and to preformat the logs.

Important: You should use the newest version of CSQ4BSDS, or update your JCL manually to use RECORDS(850 60).

The started task procedure, CSQ4MSTR, described in "Task 6: Create procedures for the IBM MQ queue manager" on page 537, refers to BSDSs in statements of the form:

```
//BSDS1   DD DSN=++HLQ++.BSDS01,DISP=SHR
//BSDS2   DD DSN=++HLQ++.BSDS02,DISP=SHR
```

The log data sets are referred to by the BSDSs.

Note:

1. The BLKSIZE must be specified on the SYSPRINT DD statement in the LOGDEF step. The BLKSIZE must be 629.
2. To help identify bootstrap data sets and log data sets from different queue managers, include the subsystem name in the high level qualifier of these data sets.

3. If you are using queue-sharing groups, you must define the bootstrap and log data sets with SHAREOPTIONS(2 3).

See Planning on z/OS for information about planning bootstrap and log data sets and their sizes.

For IBM MQ 8.0, the 8 byte log RBA enhancement improves the availability of a queue manager, as described in Larger log Relative Byte Address. To enable 8 byte log RBA on a queue manager before the queue manager is first started, perform the following steps after creating your logging environment.

1. Using **IDCAMS ALTER**, rename the version 1 format BSDSs (created using the CSQJU003 program) to something like `++HLQ++.V1.BSDS01`.

   Note: Ensure that you rename the data and index components as well as the VSAM cluster.
2. Allocate new BSDSs with the same attributes as the ones already defined. These will become the version 2 format BSDSs that will be used by the queue manager when it is started.
3. Run the BSDS conversion utility (CSQJUCNV) to convert the version 1 format BSDSs to the new version 2 format BSDSs.
4. Once the conversion completes successfully, delete the version 1 format BSDSs.
5. In order to use 8 byte log RBA, ensure that Version 8.0 new functions are enabled with OPMODE as described in "Task 17: Tailor your system parameter module" on page 548.

Note: If the queue manager is in a queue-sharing group, all queue managers in the queue-sharing group must have been started with OPMODE(NEWFUNC,800) before 8 byte log RBA can be enabled.

Related concepts

"Task 15: Define your page sets" on page 547
Define page sets for each queue manager using one of the supplied samples.

Task 15: Define your page sets

Define page sets for each queue manager using one of the supplied samples.

- *Repeat this task for each IBM MQ queue manager.*
- *You do not need to perform this task when migrating from a previous version.*

Define separate page sets for each IBM MQ queue manager. thlqual.SCSQPROC(CSQ4PAGE) and thlqual.SCSQPROC(CSQ4PAGR) contain JCL and AMS control statements to define and format page sets. Member CSQ4PAGE uses one page set for each class of message, member CSQ4PAGR uses multiple page sets for the major classes of message. The JCL runs the supplied utility program CSQUTIL. Review the samples and customize them for the number of page sets you want and the sizes to use. See the Planning on z/OS for information about page sets and how to calculate suitable sizes.

The started task procedure CSQ4MSTR described in "Task 6: Create procedures for the IBM MQ queue manager" on page 537 refers to the page sets, in a statement of the form:

```
//CSQP00 nn DD DISP=OLD,DSN= xxxxxxxxxx
```

where *nn* is the page set number between 00 and 99, and *xxxxxxxxxx* is the data set that you define.

Note:

1. If you intend to use the dynamic page set expansion feature, ensure that secondary extents are defined for each page set. thlqual.SCSQPROC(CSQ4PAGE) shows how to do this.
2. To help identify page sets from different queue managers, include the subsystem name in the high level qualifier of the data set associated with each page set.
3. If you intend to allow the FORCE option to be used with the FORMAT function of the utility program CSQUTIL, you must add the REUSE attribute on the AMS DEFINE CLUSTER statement. This is described in the Administering IBM MQ for z/OS.

4. If your page sets are to be larger than 4 GB you must use the Storage Management System (SMS) EXTENDED ADDRESSABILITY function.

Related concepts

"Task 16: Add the IBM MQ entries to the Db2 tables" on page 548
If you are using queue-sharing groups, run the CSQ5PQSG utility to add queue-sharing group and queue manager entries to the IBM MQ tables in the Db2 data-sharing group.

Task 16: Add the IBM MQ entries to the Db2 tables

If you are using queue-sharing groups, run the CSQ5PQSG utility to add queue-sharing group and queue manager entries to the IBM MQ tables in the Db2 data-sharing group.

- *Repeat this task for each IBM MQ queue-sharing group and each queue manager.*
- *You might need to perform this task when migrating from a previous version.*
- *Omit this task if you are not using queue-sharing groups.*

  If you later want to use queue-sharing groups, perform this task at that time.

Run CSQ5PQSG for each queue-sharing group and each queue manager that is to be a member of a queue-sharing group. (CSQ5PQSG is described in the Administering IBM MQ for z/OS.)

Perform the following actions in the specified order:

1. Add a queue-sharing group entry into the IBM MQ Db2 tables using the ADD QSG function of the CSQ5PQSG program. A sample is provided in thlqual.SCSQPROC(CSQ45AQS).

   Perform this function once for each queue-sharing group that is defined in the Db2 data-sharing group. The queue-sharing group entry must exist before adding any queue manager entries that reference the queue-sharing group.
2. Add a queue manager entry into the IBM MQ Db2 tables using the ADD QMGR function of the CSQ5PQSG program. A sample is provided in thlqual.SCSQPROC(CSQ45AQM).

   Perform this function for each queue manager that is to be a member of the queue-sharing group.

   Note:

   a. A queue manager can only be a member of one queue-sharing group.
   b. You must have RRS running to be able to use queue-sharing groups.

Related concepts

"Task 17: Tailor your system parameter module" on page 548
The IBM MQ system parameter module controls the logging, archiving, tracing, and connection environments that IBM MQ uses in its operation. A default module is supplied, or you can create your own using supplied JCL and assembler source modules.

Task 17: Tailor your system parameter module

The IBM MQ system parameter module controls the logging, archiving, tracing, and connection environments that IBM MQ uses in its operation. A default module is supplied, or you can create your own using supplied JCL and assembler source modules.

- *Repeat this task for each IBM MQ queue manager, as required.*
- *You need to perform this task when migrating from a previous version. For details, see* Migrating and upgrading IBM MQ.
- *To enable IBM MQ Advanced Message Security for z/OS on an existing queue manager, you only need to set SPLCAP to YES as described in* "Using CSQ6SYSP" on page 550*. If you are configuring this queue manager for the first time, complete the whole of this task.*

The system parameter module has three macros as follows:

Macro name	Purpose
CSQ6SYSP	Specifies the connection and tracing parameters, see "Using CSQ6SYSP" on page 550
CSQ6LOGP	Controls log initialization, see "Using CSQ6LOGP" on page 559
CSQ6ARVP	Controls archive initialization, see "Using CSQ6ARVP" on page 563

IBM MQ supplies a default system parameter module, CSQZPARM, which is invoked automatically if you issue the START QMGR command (without a PARM parameter) to start an instance of IBM MQ. CSQZPARM is in the APF-authorized library thlqual.SCSQAUTH also supplied with IBM MQ. The values of these parameters are displayed as a series of messages when you start IBM MQ.

See START QMGR for more information about how this command is used.

Creating your own system parameter module

If CSQZPARM does not contain the system parameters you want, you can create your own system parameter module using the sample JCL provided in thlqual.SCSQPROC(CSQ4ZPRM).

To create your own system parameter module:

1. Make a working copy of the JCL sample.
2. Edit the parameters for each macro in the copy as required. If you remove any parameters from the macro calls, the default values are automatically picked up at run time.
3. Replace the placeholder ++NAME++ with the name that the load module is to take (this can be CSQZPARM).
4. If your assembler is not high-level assembler, change the JCL as required by your assembler.
5. Run the JCL to assemble and link edit the tailored versions of the system parameter macros to produce a load module. This is the new system parameter module with the name that you have specified.
6. Put the load module produced in an APF-authorized user library.
7. Add user READ access to the APF-authorized user library.
8. Include this library in the IBM MQ queue manager started task procedure STEPLIB. This library name must come before the library thlqual.SCSQAUTH in STEPLIB.
9. Invoke the new system parameter module when you start the queue manager. For example, if the new module is named NEWMODS, issue the command:

   ```
   START QMGR PARM(NEWMODS)
   ```
10. Ensure successful completion of the command by checking the job log. There should be an entry in the log similar to the following:

    ```
    CSQ9022I CDL1 CSQYASCP 'START QMGR' NORMAL COMPLETION
    ```

You can also specify the parameter module name in the queue manager startup JCL. For more information, see Starting and stopping a queue manager.

Note: If you choose to name your module CSQZPARM, you do not need to specify the PARM parameter on the START QMGR command.

Fine tuning a system parameter module

IBM MQ also supplies a set of three assembler source modules, which can be used to fine-tune an existing system parameter module. These modules are in library thlqual.SCSQASMS. Typically, you use these modules in a test environment to change the default parameters in the system parameter macros. Each source module calls a different system parameter macro:

This assembler source module...	Calls this macro...
CSQFSYSP	CSQ6SYSP (connection and tracing parameters)
CSQJLOGP	CSQ6LOGP (log initialization)
CSQJARVP	CSQ6ARVP (archive initialization)

This is how you use these modules:

1. Make working copies of each assembler source module in a user assembler library.
2. Edit your copies by adding or altering the values of any parameters as required.
3. Assemble your copies of any edited modules to create object modules in a user object library.
4. Link edit these object code modules with an existing system parameter module to produce a load module that is the new system parameter module.
5. Ensure that new system parameter module is a member of a user authorized library.
6. Include this library in the queue manager started task procedure STEPLIB. This library must come before the library thlqual.SCSQAUTH in STEPLIB.
7. Invoke the new system parameter module by issuing a START QMGR command, specifying the new module name in the PARM parameter, as before.

A sample usermod is provided in member CSQ4UZPR of SCSQPROC which demonstrates how to manage customized system parameters under SMP/E control.

Altering system parameters

You can alter some system parameters while a queue manager is running; see the SET SYSTEM, SET LOG, and SET ARCHIVE commands.

Put the SET commands in your initialization input data sets so that they take effect every time you start the queue manager.

Related concepts

"Task 18: Tailor the channel initiator parameters" on page 570
Use ALTER QMGR to customize the channel initiator to suit your requirements.

Using CSQ6SYSP

Use this topic as a reference for how to set system parameters using CSQ6SYSP.

The default parameters for CSQ6SYSP, and whether you can alter each parameter using the SET SYSTEM command, are shown in Table 43 on page 550. If you want to change any of these values, see the detailed descriptions of the parameters.

Table 43. Default values of CSQ6SYSP parameters

Parameter	Description	Default value	SET command
ACELIM	Size of ACE storage pool in 1 KB blocks.	0 (no limit)	✓
CLCACHE	Specifies the type of cluster cache to use.	STATIC	-
CMDUSER	The default user ID for command security checks.	CSQOPR	-
CONNSWAP	Specifies whether jobs that are issuing certain IBM MQ API calls are swappable or non-swappable.	YES	-

Table 43. Default values of CSQ6SYSP parameters (continued)

Parameter	Description	Default value	SET command
EXCLMSG	Specifies a list of messages to be excluded from any log. Messages in this list are not sent to the z/OS console and hardcopy log. As a result using the EXCLMSG parameter to exclude messages is more efficient from a CPU perspective than using the methods described in “Task 22: Suppress information messages” on page 574	( )	✓
EXITLIM	Time (in seconds) for which queue-manager exits can run during each invocation.	30	-
EXITTCB	How many started server tasks to use to run queue manager exits.	8	-
LOGLOAD	Number of log records written by IBM MQ between the start of one checkpoint and the next.	500 000	✓
MULCCAPT	Determines the Measured Usage Pricing property which controls the algorithm for gathering data used by Measured Usage License Charging (MULC).	See parameter description	-
OPMODE	Controls the operation mode of the queue manager.	See parameter description	-
OTMACON	OTMA connection parameters.	See parameter description	-
QINDXBLD	Determines whether queue manager restart waits until all indexes are rebuilt, or completes before all indexes are rebuilt.	WAIT	-
QMCCSID	Coded character set identifier for the queue manager.	Zero	-
QSGDATA	Queue-sharing group parameters.	See parameter description	-
RESAUDIT	RESLEVEL auditing parameter.	YES	-
ROUTCDE	Message routing code assigned to messages not solicited from a specific console.	1	-
SERVICE	Reserved for use by IBM.	0	✓
SMFACCT	Specifies whether SMF accounting data is to be collected when the queue manager is started. Note that class 4 channel accounting data is collected only when the channel initiator is started.	NO	-

Table 43. Default values of CSQ6SYSP parameters (continued)			
Parameter	**Description**	**Default value**	**SET command**
SMFSTAT	Specifies whether SMF statistics are to be collected when the queue manager is started. Note that class 4 channel initiator statistics data is collected only when the channel initiator is started.	NO	-
SPLCAP	Specifies whether queue security policy capability is enabled on this queue manager. For IBM MQ Advanced Message Security for z/OS, set this parameter to YES.	NO	-
STATIME	Default time, in minutes, between each gathering of statistics.	30	✓
TRACSTR	Specifies whether tracing is to be started automatically.	NO	-
TRACTBL	Size of trace table, in 4 KB blocks, to be used by the global trace facility.	99 (396 KB)	✓
WLMTIME	Time between scanning the queue index for WLM-managed queues.	30	-
WLMTIMU	Units (minutes or seconds) for WLMTIME.	MINS	-

ACELIM

Specifies the maximum size of the ACE storage pool in 1 KB blocks. The number must be in the range 0-999999. The default value of zero means no imposed constraint, beyond what is available in the system.

You should only set a value for ACELIM on queue managers that have been identified as using exorbitant quantities of ECSA storage. Limiting the ACE storage pool has the effect of limiting the number of connections in the system, and so, the amount of ECSA storage used by a queue manager.

Once the queue manager reaches the limit it is not possible for applications to obtain new connections. The lack of new connections causes failures in MQCONN processing, and applications coordinated through RRS are likely to experience failures in any IBM MQ API.

An ACE represents approximately 12.5% of the total ECSA required for the thread-related control blocks for a connection. So, for example, specifying ACELIM=5120 would be expected to cap the total amount of ECSA allocated by the queue manager (for thread-related control blocks) at approximately 40960K; that is 5120 multiplied by 8.

In order to cap the amount total amount of ECSA allocated by the queue-manager, for thread-related control blocks at 5120K, an ACELIM value of 640 is required.

You can use SMF 115 subtype 5 records, produced by statistics CLASS(3) trace, to monitor the size of the 'ACE/PEB' storage pool, and hence set an appropriate value for ACELIM.

You can obtain the total amount of ECSA storage used by the queue-manager, for control blocks, from SMF 115 subtype 7 records, written by statistics CLASS(2) trace; that is the first two elements in QSRSPHBT added together.

Note that, you should consider setting ACELIM as a mechanism to protect a z/OS image from a badly behaving queue manager, rather than as a means to control application connections to a queue manager.

CLCACHE
Specifies the type of cluster cache to use. See "Configuring a queue manager cluster" on page 215 for more information.

STATIC
When the cluster cache is static, its size is fixed at queue manager start-up, enough for the current amount of cluster information plus some space for expansion. The size cannot increase while the queue manager is active. This is the default.

DYNAMIC
When the cluster cache is dynamic, the initial size allocated at queue manager startup can be increased automatically if required while the queue manager is active.

CMDUSER
Specifies the default user ID used for command security checks. This user ID must be defined to the ESM (for example, RACF). Specify a name of 1 through 8 alphanumeric characters. The first character must be alphabetic.

The default is CSQOPR.

CONNSWAP
Specifies whether batch jobs that are issuing certain IBM MQ API calls are swappable or non-swappable for the duration of the IBM MQ API request. Specify one of the following values:

NO
Jobs are non-swappable during certain IBM MQ API calls.

YES
Jobs are swappable during all IBM MQ API calls.

The default value is YES.

Use this parameter if low-priority jobs are swapped out while holding IBM MQ resources that other jobs or tasks might be waiting for. This parameter replaces the service parameter that was included IBM MQ V701; the service parameter is no longer in use.

IBM MQ views WebSphere Application Server as part of an RRSBATCH environment. From IBM WebSphere MQ 7.1, when the CONNSWAP keyword is used, it is applied to any application in a BATCH or RRSBATCH environment. The CONNSWAP keyword is also applicable to TSO users, however, it is not applicable for CICS or IMS applications. CONNSWAP changes are implemented when a recycle of the queue manager takes place. A recycle is required after the keyword change is made, because the CSQ6SYSP macro is reassembled, and the queue-manager restarted using the load module which is updated by the macro.

Alternatively, the WebSphere Application Server address space can be made non-swappable by using PPT.

EXCLMSG
Specifies a list of error messages to be excluded.

This list is dynamic and is updated using the SET SYSTEM command.

The default value is an empty list ( ).

Messages are supplied without the CSQ prefix and without the action code suffix (I-D-E-A). For example, to exclude message CSQX500I, add X500 to this list. This list can contain a maximum of 16 message identifiers.

To be eligible to be included in the list, the message must be issued after normal start up of the MSTR or CHIN address spaces and begin with the one of the following characters E, H, I, J, L, M, N, P, R, T, V, W, X, Y, 2 ,3, 5, 9.

Message identifiers that are issued as a result of processing commands can be added to the list, however will not be excluded. For example, a message identifier is issued as a result of the DISPLAY USAGE PSID(*) command, however, this message can not be suppressed.

EXITLIM
Specifies the time, in seconds, allowed for each invocation of the queue manager exits. (This parameter has no effect on channel exits.)

Specify a value in the range 5 through 9999.

The default is 30. The queue manager polls exits that are running every 30 seconds. On each poll, any that have been running for more than the time specified by EXITLIM are forcibly terminated.

EXITTCB
Specifies the number of started server tasks to use to run exits in the queue manager. (This parameter has no effect on channel exits.) You must specify a number at least as high as the maximum number of exits (other than channel exits) that the queue manager might have to run, else it will fail with a 6c6 abend.

Specify a value in the range zero through 99. A value of zero means that no exits can be run.

The default is 8.

LOGLOAD
Specifies the number of log records that IBM MQ MQ writes between the start of one checkpoint and the next. IBM MQ starts a new checkpoint after the number of records that you specify has been written.

Specify a value in the range 200 through 16 000 000.

The default is 500 000.

The greater the value, the better the performance of IBM MQ ; however, restart takes longer if the parameter is set to a large value.

Suggested settings:

Test system	10 000
Production system	500 000 In a production system, the supplied default value might result in a checkpoint frequency that is too high.

The value of LOGLOAD determines the frequency of queue manager checkpoints. Too large a value means that a large amount of data is written to the log between checkpoints, resulting in an increased queue manager forward recovery restart time following a failure. Too small a value causes checkpoints to occur too frequently during peak load, adversely affecting response times and processor usage.

An initial value of 500 000 is suggested for LOGLOAD. For a 1 KB persistent message rate of 100 messages a second (that is, 100 `MQPUT` s with commit and 100 `MQGET` s with commit) the interval between checkpoints is approximately 5 minutes.

Note: This is intended as a guideline only and the optimum value for this parameter is dependent on the characteristics of the individual system.

MULCCAPT
Specifies the algorithm to be used for gathering data used by Measured Usage License Charging (MULC).

STANDARD
MULC is based on the time from the IBM MQ API MQCONN call to the time of the IBM MQ API MQDISC call.

REFINED
MULC is based on the time from the start of an IBM MQ API call to the end of the IBM MQ API call.

The default is STANDARD

OPMODE=(Mode, *VerificationLevel*)

OPMODE specifies the operation mode of the queue manager.

The default setting of OPMODE is `OPMODE=(COMPAT,800)`.

Mode

Specifies the requested operation mode. The values are as follows:

COMPAT

The queue manager runs in compatibility mode. Certain new functions are not available. The queue manager can be migrated back to an earlier release.

NEWFUNC

All new functions provided in this level of code are available. The queue manager cannot be migrated back to an earlier release.

VerificationLevel

VerificationLevel is a `Version.Release.Modification` (VRM) code, without punctuation; `800`, for example.

The value of *VerificationLevel* ensures that the **CSQ6SYSP** parameters are coded for use with the level of **CSQ6SYSP** macro being compiled. If *VerificationLevel* does not match the VRM level of SCSQMACS used for **CSQ6SYSP**, then a compile-time error is reported. The *VerificationLevel* is compiled into the parameter module.

At queue manager startup, if the *VerificationLevel* does not match the release level of the queue manager, then `COMPAT` mode is forced.

The intent of the *VerificationLevel* parameter is to avoid inadvertent and irreversible setting of OPMODE to `NEWFUNC`. The mistake might occur when migrating to a newer version of IBM MQ using **CSQ6SYSP** statements prepared for an older version of the queue manager. It might also occur using a **CSQ6SYSP** parameter module built with an older version of the SCSQMACS macros.

OTMACON

OTMA parameters. This keyword takes five positional parameters::

OTMACON = (Group,Member,Druexit,Age,Tpipepfx)

Group

This is the name of the XCF group to which this particular instance of IBM MQ belongs.

It can be 1 through 8 characters long and must be entered in uppercase characters.

The default is blanks, which indicates that IBM MQ must not attempt to join an XCF group.

Member

This is the member name of this particular instance of IBM MQ within the XCF group.

It can be 1 through 16 characters long and must be entered in uppercase characters.

The default is the 4-character queue manager name.

Druexit

This specifies the name of the OTMA destination resolution user exit to be run by IMS.

It can be 1 through 8 characters long.

The default is DFSYDRU0.

This parameter is optional; it is required if IBM MQ is to receive messages from an IMS application that was not started by IBM MQ. The name must correspond to the destination resolution user exit coded in the IMS system. For more information see "Using OTMA exits in IMS" on page 626.

Age

This represents the length of time, in seconds, that a user ID from IBM MQ is considered previously verified by IMS.

It can be in the range zero through 2 147 483 647.

The default is 2 147 483 647.

You are recommended to set this parameter in conjunction with the `interval` parameter of the ALTER SECURITY command to maintain consistency of security cache settings across the mainframe.

Tpipepfx
This represents the prefix to be used for Tpipe names.

It comprises three characters; the first character is in the range A through Z, subsequent characters are A through Z or 0 through 9. The default is CSQ.

This is used each time IBM MQ creates a Tpipe; the rest of the name is assigned by IBM MQ. You cannot set the full Tpipe name for any Tpipe created by IBM MQ.

QINDXBLD
Determines whether queue manager restart waits until all queue indexes are rebuilt, or completes before all indexes are rebuilt.

WAIT
Queue manager restart waits for all queue index builds to be completed. This means that no applications are delayed during normal IBM MQ API processing while the index is created, as all indexes are created before any applications can connect to the queue manager.

This is the default.

NOWAIT
The queue manager can restart before all queue index building is completed.

QMCCSID
Specifies the default coded character set identifier that the queue manager (and therefore distributed queuing) is to use.

Specify a value in the range zero through 65535. The value must represent an EBCDIC codepage listed as a native z/OS codepage for your chosen language in National languages.

Zero, which is the default value, means use the CCSID currently set or, if none is set, use CCSID 500. This means that if you have explicitly set the CCSID to any non-zero value, you cannot reset it by setting QMCCSID to zero; you must now use the correct non-zero CCSID. If QMCCSID is zero, you can check what CCSID is actually in use by issuing the command DISPLAY QMGR CCSID.

On distributed platforms, use the ALTER QMGR command.

QSGDATA
Queue-sharing group data. This keyword takes five positional parameters:

QSGDATA=(`Qsgname,Dsgname,Db2name,Db2serv ,Db2blob`)

Qsgname
This is the name of the queue-sharing group to which the queue manager belongs.

See Rules for naming IBM MQ objects for valid characters. The name:

- Can be 1 through 4 characters long
- Must not start with a numeric
- Must not end in @.

  This is because, for implementation reasons, names of less than four characters are padded internally with @ symbols,

The default is blanks, which indicates that the queue manager is not a member of any queue-sharing group.

Dsgname
This is the name of the Db2 data-sharing group to which the queue manager is to connect.

It can be 1 through 8 characters long and must be entered in uppercase characters.

The default is blanks, which indicates that you are not using queue-sharing groups.

Db2name
This is the name of the Db2 subsystem or group attachment to which the queue manager is to connect.

It can be 1 through 4 characters long and must be entered in uppercase characters.

The default is blanks, which indicates that you are not using queue-sharing groups.

Note: The Db2 subsystem (or group attachment) must be in the Db2 data-sharing group specified in the `Dsgname`, and all queue managers must specify the same Db2 data-sharing group.

Db2serv
This is the number of server tasks used for accessing Db2.

It can be in the range 4 through 10.

The default is 4.

Db2blob
This is the number of Db2 tasks used for accessing Binary Large Objects (BLOBs).

It can be in the range 4 through 10.

The default is 4.

If you specify one of the name parameters (that is, **`Qsgname`**, **`Dsgname`**, or **`Db2name`**), you must enter values for the other names, otherwise IBM MQ fails.

RESAUDIT
Specifies whether RACF audit records are written for RESLEVEL security checks performed during connection processing.

Specify one of:

NO
RESLEVEL auditing is not performed.

YES
RESLEVEL auditing is performed.

The default is YES.

ROUTCDE
Specifies the default z/OS message routing code assigned to messages that are not sent in direct response to an MQSC command.

Specify one of:

1. A value in the range 1 through 16, inclusive.
2. A list of values, separated by a comma and enclosed in parentheses. Each value must be in the range 1 through 16, inclusive.

The default is 1.

For more information about z/OS routing codes, see the *MVS Routing and Descriptor Codes* manual.

SERVICE
This field is reserved for use by IBM.

SMFACCT
Specifies whether IBM MQ sends accounting data to SMF automatically when the queue manager starts.

Specify one of:

NO
Do not start gathering accounting data automatically.

YES
Start gathering accounting data automatically for the default class 1.

integers
A list of classes for which accounting is gathered automatically in the range 1 through 4.

The default is NO.

SMFSTAT
Specifies whether to gather SMF statistics automatically when the queue manager starts.

Specify one of:

NO
Do not start gathering statistics automatically.

YES
Start gathering statistics automatically for the default class 1.

integers
A list of classes for which statistics are gathered automatically in the range 1 through 4.

The default is NO.

SPLCAP
The security policy capability enables higher level of message security through policies that control whether messages are signed, or encrypted, as they are written and read from queues.

Its use is licensed by a separately installed product, IBM MQ Advanced Message Security (AMS), which supplies an enabling module in the SDRQAUTH library.

Security policy processing is enabled for this queue manager, by configuring SPLCAP with one of the following values:

NO
The capability to implement message security policies for queues is not enabled during queue manager initialization.

YES
Message security capabilities are enabled during queue manager initialization.

If this control is set, the queue manager attempts to load the license enabling module from SDRQAUTH during initialization, and start an additional address space (AMSM).

The queue manager does not start unless AMS is licensed, and the necessary configuration for message security is in place.

The default is NO.

STATIME
Specifies the default time, in minutes, between consecutive gatherings of statistics.

Specify a number in the range zero through 1440.

If you specify a value of zero, both statistics data and accounting data is collected at the SMF data collection broadcast. See Using System Management Facility for information about setting this.

The default is 30.

TRACSTR
Specifies whether global tracing is to start automatically.

Specify one of:

NO
Do not start global tracing automatically.

YES
Start global tracing automatically for the default class, class 1.

integers
A list of classes for which global tracing is to be started automatically in the range 1 through 4.

*
Start global trace automatically for all classes.

The default is NO if you do not specify the keyword in the macro.

Note: The supplied default system parameter load module (CSQZPARM) has TRACSTR=YES (set in the assembler module CSQFSYSP). If you do not want to start tracing automatically, either create your own system parameter module, or issue the STOP TRACE command after the queue manager has started.

For details about the STOP TRACE command, see STOP TRACE.

TRACTBL
Specifies the default size, in 4 KB blocks, of trace table where the global trace facility stores IBM MQ trace records.

Specify a value in the range 1 through 999.

The default is 99. This is equivalent to 396 KB.

Note: Storage for the trace table is allocated in the ECSA. Therefore, you must select this value with care.

WLMTIME
Specifies the time (in minutes or seconds depending on the value of WLMTIMU) between each scan of the indexes for WLM-managed queues.

Specify a value in the range 1 through 9999.

The default is 30.

WLMTIMU
Time units used with the WLMTIME parameter.

Specify one of :

MINS
WLMTIME represents a number of minutes.

SECS
WLMTIME represents a number of seconds.

The default is MINS.

Related reference

"Using CSQ6LOGP" on page 559
Use this topic as a reference for how to specify logging options using CSQ6LOGP.

"Using CSQ6ARVP" on page 563
Use this topic as a reference for how to specify your archiving environment using CSQ6ARVP

Using CSQ6LOGP

Use this topic as a reference for how to specify logging options using CSQ6LOGP.

Use CSQ6LOGP to establish your logging options.

The default parameters for CSQ6LOGP, and whether you can alter each parameter using the SET LOG command, are shown in Default values of CSQ6LOGP parameters. If you need to change any of these values, refer to the detailed descriptions of the parameters.

Table 44. Default values of CSQ6LOGP parameters

Parameter	Description	Default value	SET command
COMPLOG	Controls whether log compression is enabled.	NONE	X

Table 44. Default values of CSQ6LOGP parameters (continued)			
Parameter	**Description**	**Default value**	**SET command**
DEALLCT	Length of time an archive tape unit remains unused before it is deallocated.	zero	X
INBUFF	Size of input buffer storage for active and archive log data sets.	60 KB	-
MAXARCH	Maximum number of archive log volumes that can be recorded.	500	X
MAXCNOFF	Maximum number of CSQJOFF7 offload tasks that can be run in parallel.	31	-
MAXRTU	Maximum number of dedicated tape units allocated to read archive log tape volumes concurrently.	2	X
OFFLOAD	Archiving on or off.	YES (ON)	-
OUTBUFF	Size of output buffer storage for active and archive log data sets.	4 000 KB	-
TWOACTV	Single or dual active logging.	YES (dual)	-
TWOARCH	Single or dual archive logging.	YES (dual)	-
TWOBSDS	Single or dual BSDS.	YES (dual BSDS)	-
WRTHRSH	Number of output buffers to be filled before they are written to the active log data sets.	20	X

COMPLOG

Specifies whether log compression is enabled.

Specify either:

NONE

Log compression is not enabled.

RLE

Log compression is enabled using run-length encoding.

ANY

The queue manager selects the compression algorithm that gives the greatest degree of log record compression. This option results in RLE compression.

The default is NONE.

For more details about log compression, see Log compression.

DEALLCT

Specifies the length of time, in minutes, that an archive read tape unit is allowed to remain unused before it is deallocated.

Specify one of the following:

- Time, in minutes, in the range zero through 1440
- NOLIMIT

Specifying 1440 or NOLIMIT means that the tape unit is never deallocated.

The default is zero.

When archive log data is being read from tape, it is recommended that you set this value high enough to allow IBM MQ to optimize tape handling for multiple read applications.

INBUFF

Specifies the size, in kilobytes, of the input buffer for reading the active and archive logs during recovery. Use a decimal number in the range 28 through 60. The value specified is rounded up to a multiple of 4.

The default is 60 KB.

Suggested settings:

Test system	28 KB
Production system	60 KB Set this to the maximum for best log read performance.

MAXARCH

Specifies the maximum number of archive log volumes that can be recorded in the BSDS. When this number is exceeded, recording begins again at the start of the BSDS.

Use a decimal number in the range 10 through 1000.

The default is 500.

Suggested settings:

Test system	500 (default)
Production system	1 000 Set this to the maximum so that the BSDS can record as many logs as possible.

For information about the logs and BSDS, see Managing IBM MQ resources.

MAXCNOFF

Specifies the number of CSQJOFF7 offload tasks that can be run in parallel.

This allows a queue manager, or queue managers, to be tuned such that they will not use all the available tape units.

Instead the queue manager waits until a CSQJOFF7 offload task has completed before trying to allocate any new archive data sets.

If the queue manager is archiving to tape, set this parameter so that the number of concurrent tape requests should not equal, or exceed, the number of tape units available, otherwise the system might hang.

Note that if dual archiving is in use, then each offload task performs both archives, so the parameter needs to be set accordingly. For example if the queue manager is dual archiving to tape, a value of MAXCNOFF=2 would allow up to two active logs to be archived concurrently to four tapes.

If several queue managers are sharing the tape units, you should set the MAXCNOFF for each queue manager accordingly.

The default value is 31.

Specify a value in the range 1 through 31.

MAXRTU

Specifies the maximum number of dedicated tape units that can be allocated to read archive log tape volumes concurrently.

This parameter and the DEALLCT parameter allow IBM MQ to optimize archive log reading from tape devices.

Specify a value in the range 1 through 99.

The default is 2.

It is recommended that you set the value to be at least one less than the number of tape units available to IBM MQ. If you do otherwise, the offload process could be delayed, which could affect the performance of your system. For maximum throughput during archive log processing, specify the largest value possible for this option, remembering that you need at least one tape unit for offload processing.

OFFLOAD
Specifies whether archiving is on or off.

Specify either:

YES
Archiving is on

NO
Archiving is off

The default is YES.

Attention: Do **not** switch archiving off unless you are working in a test environment. If you do switch it off, you cannot guarantee that data will be recovered in the event of a system or transaction failure.

OUTBUFF
Specifies the total size, in kilobytes, of the storage to be used by IBM MQ for output buffers for writing the active and archive log data sets. Each output buffer is 4 KB.

The parameter must be in the range 80 through 4000. The value specified is rounded up to a multiple of 4. Values between 40and 80 will be accepted for compatibility reasons, and are treated as a value of 80.

The default is 4 000 KB.

Suggested settings:

Test system	400 KB
Production system	4 000 KB Set this value to the maximum to avoid running out of log output buffers.

TWOACTV
Specifies single or dual active logging.

Specify either:

NO
Single active logs

YES
Dual active logs

The default is YES.

For more information about the use of single and dual logging, see Managing IBM MQ resources.

TWOARCH
Specifies the number of archive logs that IBM MQ produces when the active log is offloaded.

Specify either:

NO
Single archive logs

YES
Dual archive logs

The default is YES.

Suggested settings:

Test system	NO
Production system	YES (default)

For more information about the use of single and dual logging, see Managing IBM MQ resources.

TWOBSDS
Specifies the number of bootstrap data sets.

Specify either:

NO
Single BSDS

YES
Dual BSDS

The default is YES.

For more information about the use of single and dual logging, see Managing IBM MQ resources.

WRTHRSH
Specifies the number of 4 KB output buffers to be filled before they are written to the active log data sets.

The larger the number of buffers, the less often the write takes place, and this improves the performance of IBM MQ. The buffers might be written before this number is reached if significant events, such as a commit point, occur.

Specify the number of buffers in the range 1 through 256.

The default is 20.

Related reference

"Using CSQ6SYSP" on page 550
Use this topic as a reference for how to set system parameters using CSQ6SYSP.

"Using CSQ6ARVP" on page 563
Use this topic as a reference for how to specify your archiving environment using CSQ6ARVP

Using CSQ6ARVP

Use this topic as a reference for how to specify your archiving environment using CSQ6ARVP

Use CSQ6ARVP to establish your archiving environment.

The default parameters for CSQ6ARVP, and whether you can alter each parameter using the SET ARCHIVE command, are shown in Table 45 on page 563.

If free space on the archive DASD volumes is likely to be fragmented, you are recommended to specify a smaller primary extent and allow expansion into secondary extents. For more information about space allocation for active logs, refer to Planning your log archive storage.

Table 45. Default values of CSQ6ARVP parameters

Parameter	Description	Default value	SET command
ALCUNIT	Units in which primary and secondary space allocations are made.	BLK (blocks)	X
ARCPFX1	Prefix for first archive log data set name.	CSQARC1	X
ARCPFX2	Prefix for second archive log data set name.	CSQARC2	X

Table 45. Default values of CSQ6ARVP parameters (continued)			
Parameter	**Description**	**Default value**	**SET command**
ARCRETN	The retention period of the archive log data set in days.	9999	X
ARCWRTC	List of route codes for messages to the operator about archive log data sets.	1,3,4	X
ARCWTOR	Whether to send message to operator and wait for reply before trying to mount an archive log data set.	YES	X
BLKSIZE	Block size of archive log data set.	28 672	X
CATALOG	Whether archive log data sets are cataloged in the ICF.	NO	X
COMPACT	Whether archive log data sets should be compacted.	NO	X
PRIQTY	Primary space allocation for DASD data sets.	25 715	X
PROTECT	Whether archive log data sets are protected by ESM profiles when the data sets are created.	NO	X
QUIESCE	Maximum time, in seconds, allowed for quiesce when ARCHIVE LOG with MODE(QUIESCE) specified.	5	X
SECQTY	Secondary space allocation for DASD data sets. See the ALCUNIT parameter for the units to be used.	540	X
TSTAMP	Whether the archive data set name should include a time stamp.	NO	X
UNIT	Device type or unit name on which the first copy of archive log data sets is stored.	TAPE	X
UNIT2	Device type or unit name on which the second copy of archive log data sets is stored.	Blank	X

ALCUNIT

Specifies the unit in which primary and secondary space allocations are made.

Specify one of:

CYL

Cylinders

TRK

Tracks

BLK

Blocks

You are recommended to use BLK because it is independent of the device type.

The default is BLK.

If free space on the archive DASD volumes is likely to be fragmented, you are recommended to specify a smaller primary extent and allow expansion into secondary extents. For more information about space allocation for active logs, refer to the Planning on z/OS.

ARCPFX1

Specifies the prefix for the first archive log data set name.

See the TSTAMP parameter for a description of how the data sets are named and for restrictions on the length of ARCPFX1.

This parameter cannot be left blank.

The default is CSQARC1.

You might need to authorize the userid associated with the IBM MQ queue manager address space to create archive logs with this prefix.

ARCPFX2
Specifies the prefix for the second archive log data set name.

See the TSTAMP parameter for a description of how the data sets are named and for restrictions on the length of ARCPFX2.

This parameter cannot be blank even if the TWOARCH parameter is specified as NO.

The default is CSQARC2.

You might need to authorize the userid associated with the IBM MQ queue manager address space to create archive logs with this prefix.

ARCRETN
Specifies the retention period, in days, to be used when the archive log data set is created.

The parameter must be in the range zero through 9999.

The default is 9999.

Suggested settings:

Test system	3 In a test system, archive logs are probably not required over long periods.
Production system	9 999 (default) Set this value high to effectively switch automatic archive log deletion off.

For more information about discarding archive log data sets, see Discarding archive log data sets.

ARCWRTC
Specifies the list of z/OS routing codes for messages about the archive log data sets to the operator. This field is ignored if ARCWTOR is set to NO.

Specify up to 14 routing codes, each with a value in the range 1 through 16. You must specify at least one code. Separate codes in the list by commas, not by blanks.

The default is the list of values: 1,3,4.

For more information about z/OS routing codes, see the *MVS Routing and Descriptor Codes* manual.

ARCWTOR
Specifies whether a message is to be sent to the operator and a reply is received before attempting to mount an archive log data set.

Other IBM MQ users might be forced to wait until the data set is mounted, but they are not affected while IBM MQ is waiting for the reply to the message.

Specify either:

YES
The device needs a long time to mount archive log data sets. For example, a tape drive.

NO
The device does not have long delays. For example, DASD.

The default is YES.

Suggested settings:

Test system	NO
Production system	YES (default) This is dependent on operational procedures. If tape robots are used, NO might be more appropriate.

BLKSIZE

Specifies the block size of the archive log data set. The block size you specify must be compatible with the device type you specify in the UNIT parameter.

The parameter must be in the range 4 097 through 28 672. The value you specify is rounded up to a multiple of 4 096.

The default is 28 672.

This parameter is overridden by the storage management subsystem (SMS) data class blocksize, if it is provided

If the archive log data set is written to DASD, you are recommended to choose the maximum block size that allows 2 blocks for each track. For example, for a 3390 device, you should use a block size of 24 576.

If the archive log data set is written to tape, specifying the largest possible block size improves the speed of reading the archive log. You should use a block size of 28 672.

Suggested settings:

Test system	Use the block size recommendation depending on the media used for archive logs. That is, for disk 24 576, and tape 28 672.
Production system	Use the block size recommendation depending on the media used for archive logs. That is, for disk 24 576, and tape 28 672.

CATALOG

Specifies whether archive log data sets are cataloged in the primary integrated catalog facility (ICF) catalog.

Specify either:

NO

Archive log data sets are not cataloged

YES

Archive log data sets are cataloged

The default is NO.

All archive log data sets allocated on DASD must be cataloged. If you archive to DASD with the CATALOG parameter set to NO, message CSQJ072E is displayed each time an archive log data set is allocated, and IBM MQ catalogs the data set.

Suggested settings:

Test system	YES
Production system	YES, when archives are allocated on DASD

COMPACT

Specifies whether data written to archive logs is to be compacted. This option applies only to a 3480 or 3490 device that has the improved data recording capability (IDRC) feature. When this feature is turned on, hardware in the tape control unit writes data at a much higher density than normal, allowing for more data on each volume. Specify NO if you do not use a 3480 device with the IDRC feature or a 3490 base model, except for the 3490E. Specify YES if you want the data to be compacted.

Specify either:

NO
Do not compact the data sets

YES
Compact the data sets

The default is NO.

Specifying YES adversely affects performance. Also be aware that data compressed to tape can be read only using a device that supports the IDRC feature. This can be a concern if you have to send archive tapes to another site for remote recovery.

Suggested settings:

Test system	Not applicable
Production system	NO (default) This applies to 3480 and 3490 IDR compression only. Setting this to YES might degrade archive log read performance during recovery and restart; however, it does not affect writing to tape.

PRIQTY
Specifies the primary space allocation for DASD data sets in ALCUNITs.

The value must be greater than zero.

The default is 25 715.

This value must be sufficient for a copy of either the log data set or its corresponding BSDS, whichever is the larger. To determine the necessary value, follow this procedure:

1. Determine the number of active log records allocated (c) as explained in “Task 14: Create the bootstrap and log data sets” on page 546.
2. Determine the number of 4096 byte blocks in each archive log block:

```
d = BLKSIZE / 4096
```

   where BLKSIZE is the rounded up value.
3. If ALCUNIT=BLK:

```
PRIQTY = INT(c / d) + 1
```

   where INT means round down to an integer.

   If ALCUNIT=TRK:

```
PRIQTY = INT(c / (d * INT(e/BLKSIZE))) + 1
```

   where e is the number of bytes for each track (56664 for a 3390 device) and INT means round down to an integer.

If ALCUNIT=CYL:

```
PRIQTY = INT(c / (d * INT(e/BLKSIZE) * f)) + 1
```

where f is the number of tracks for each cylinder (15 for a 3390 device) and INT means round down to an integer.

For information about how large to make your log and archive data sets, see "Task 14: Create the bootstrap and log data sets" on page 546 and "Task 15: Define your page sets" on page 547.

Suggested settings:

Test system	1 680 Sufficient to hold the entire active log, that is: `10 080 / 6 = 1 680 blocks`
Production system	Not applicable when archiving to tape.

If free space on the archive DASD volumes is likely to be fragmented, you are recommended to specify a smaller primary extent and allow expansion into secondary extents. For more information about space allocation for active logs, refer to the Planning on z/OS.

PROTECT

Specifies whether archive log data sets are to be protected by discrete ESM (external security manager) profiles when the data sets are created.

Specify either:

NO

Profiles are not created.

YES

Discrete data set profiles are created when logs are offloaded. If you specify YES:

- ESM protection must be active for IBM MQ.
- The user ID associated with the IBM MQ queue manager address space must have authority to create these profiles.
- The TAPEVOL class must be active if you are archiving to tape.

Otherwise, offloading fails.

The default is NO.

QUIESCE

Specifies the maximum time in seconds allowed for the quiesce when an ARCHIVE LOG command is issued with MODE(QUIESCE) specified.

The parameter must be in the range 1 through 999.

The default is 5.

SECQTY

Specifies the secondary space allocation for DASD data sets in ALCUNITs. The secondary extent can be allocated up to 15 times; see the *z/OS MVS JCL Reference* and *z/OS MVS JCL User's Guide* for details.

The parameter must be greater than zero.

The default is 540.

TSTAMP
Specifies whether the archive log data set name has a time stamp in it.

Specify either:

NO
Names do not include a time stamp. The archive log data sets are named:

```
arcpfxi.A nnnnnnn
```

Where *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 35 characters.

YES
Names include a time stamp. The archive log data sets are named:

```
arcpfxi.cyyddd.T hhmmsst.A nnnnnnn
```

where *c* is 'D' for the years up to and including 1999 or 'E' for the year 2000 and later, and *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 19 characters.

EXT
Names include a time stamp. The archive log data sets are named:

```
arcpfxi.D yyyyddd.T hhmmsst.A nnnnnnn
```

Where *arcpfxi* is the data set name prefix specified by ARCPFX1 or ARCPFX2. *arcpfxi* can have up to 17 characters.

The default is NO.

UNIT
Specifies the device type or unit name of the device that is used to store the first copy of the archive log data set.

Specify a device type or unit name of 1 through 8 alphanumeric characters. The first character must be alphabetic.

This parameter cannot be blank.

The default is TAPE.

If you archive to DASD, you can specify a generic device type with a limited volume range, for example, UNIT=3390.

If you archive to DASD, make sure that:

- The primary space allocation is large enough to contain all the data from the active log data sets.
- The archive log data set catalog option (CATALOG) is set to YES.
- You have used a proper value for BLKSIZE.

If you archive to TAPE, IBM MQ can extend to a maximum of 20 volumes.

Suggested settings:

Test system	DASD
Production system	TAPE

For more information about choosing a location for archive logs, refer to the Planning on z/OS.

UNIT2

Specifies the device type or unit name of the device that is used to store the second copy of the archive log data sets.

Specify a device type or unit name of 1 through 8 alphanumeric characters. The first character must be alphabetic. If this parameter is blank, the value set for the UNIT parameter is used.

The default is blank.

Related reference

"Using CSQ6SYSP" on page 550
Use this topic as a reference for how to set system parameters using CSQ6SYSP.

"Using CSQ6LOGP" on page 559
Use this topic as a reference for how to specify logging options using CSQ6LOGP.

Task 18: Tailor the channel initiator parameters

Use ALTER QMGR to customize the channel initiator to suit your requirements.

- *Repeat this task for each IBM MQ queue manager, as required.*
- *You must perform this task when migrating from a previous version.*

A number of queue manager attributes control how distributed queuing operates. Set these attributes using the MQSC command ALTER QMGR. The initialization data set sample thlqual.SCSQPROC(CSQ4INYG) contains some settings that you can customize. For more information, see ALTER QMGR.

The values of these parameters are displayed as a series of messages each time you start the channel initiator.

The relationship between adapters, dispatchers, and maximum number of channels

The ALTER QMGR parameters CHIADAPS and CHIDISPS define the number of task control blocks (TCBs) used by the channel initiator. CHIADAPS (adapter) TCBs are used to make IBM MQ API calls to the queue manager. CHIDISPS (dispatcher) TCBs are used to make calls to the communications network.

The ALTER QMGR parameter MAXCHL influences the distribution of channels over the dispatcher TCBs.

CHIDISPS

If you have a small number of channels use the default value.

One task for each processor optimizes system performance. As dispatcher tasks are CPU intensive, the principle is to keep as few tasks as busy as possible, so that the time taken to find and start threads is minimized.

CHIDISPS(20) is suitable for systems with more than 100 channels. There is unlikely to be any significant disadvantage in having CHIDISPS(20) where this is more dispatcher TCBs than necessary.

As a guideline, if you have more than 1000 channels, allow one dispatcher for every 50 current channels. For example, specify CHIDISPS(40) to handle up to 2000 active channels.

If you are using TCP/IP, the maximum number of dispatchers used for TCP/IP channels is 100, even if you specify a larger value in CHIDISPS.

CHIADAPS

Each IBM MQ API call to the queue manager is independent of any other and can be made on any adapter TCB. Calls using persistent messages can take much longer than those for nonpersistent messages because of log I/O. Thus a channel initiator processing a large number of persistent messages across many channels may need more than the default 8 adapter TCBs for optimum performance. This is particularly so where achieved batchsize is small, because end of batch processing also requires log I/O, and where thin client channels are used.

The suggested value for a production environment is CHIADAPS(30). Using more than this is unlikely to give any significant extra benefit, and there is unlikely to be any significant disadvantage in having CHIADAPS(30) if this is more adapter TCBs than necessary.

MAXCHL

Each channel is associated with a particular dispatcher TCB at channel start and remains associated with that TCB until the channel stops. Many channels can share each TCB. MAXCHL is used to spread channels across the available dispatcher TCBs. The first (MIN((MAXCHL / CHIDISPS) , 10)) channels to start are associated with the first dispatcher TCB, and so on, until all dispatcher TCBs are in use.

The effect of this for small numbers of channels and a large MAXCHL is that channels are NOT evenly distributed across dispatchers. For example, if you set CHIDISPS(10) and left MAXCHL at its default value of 200 but had only 50 channels, five dispatchers would be associated with 10 channels each and five would be unused. We suggest setting MAXCHL to the number of channels actually to be used where this is a small fixed number.

If you change this queue manager property, you must also review the ACTCHL, LU62CHL, and TCPCHL queue manager properties to ensure that the values are compatible. See Queue manager parameters for a full description of these properties, and their relationship.

Setting up your z/OS UNIX System Services environment for channel initiators

The channel initiator (CHINIT) uses OMVS threads. Review the OMVS configuration parameters before creating a new CHINIT, or modifying the number of dispatchers or SSLTASKS.

Each CHINIT uses 3 + CHIDISP + SSLTASKS OMVS threads. These contribute to the total number of OMVS threads used in the LPAR, and towards the number of threads used by CHINIT started task user ID.

You can use the **`D OMVS,L`** and review the current usage, highwater usage, and system limit of MAXPROCSYS (the maximum number of processes that the system allows).

If you are adding a new CHINIT or increasing the values of CHIDISPS or SSLTASKS then you must calculate the increase in threads and review the impact on the MAXPROCSYS values. You can use the **SETOMVS** command to dynamically change the MAXPROCSYS, or update the BPXPRCxx parmlib value or both.

The OMVS parameter MAXPROCUSER is the number of OMVS threads a single OMVS user, that is with the same UID, can have. The threads count towards this value. So if you have 2 CHINITS with the same started task user ID, with 10 dispatchers and 3 SSLTASKS each then there are 2 *(3 +10 + 3) = 32 threads for the OMVS uid.

You can display the default MAXPROCUSER by issuing the **`D OMVS,O`** command and you can use the **SETOMVS** command to dynamically change the MAXPROCUSER, or update the BPXPRCxx parmlib value or both.

You can override this value on a per user basis with the RACF command **`ALTUSER userid OMVS(PROCUSERMAX(nnnn))`** or equivalent.

To start the channel initiator, issue the following command:

```
START CHINIT
```

To ensure that the channel initiator has started successfully, check that there is no ICH408I error in the xxxxCHIN(ssidCHIN) job log.

Related concepts

“Task 19: Set up Batch, TSO, and RRS adapters” on page 572
Make the adapters available to applications by adding libraries to appropriate STEPLIB concatenations. To cater for SNAP dumps issued by an adapter, allocate a CSQSNAP DDname. Consider using CSQBDEFV to improve the portability of your application programs

Channel initiator statistics data records

Task 19: Set up Batch, TSO, and RRS adapters

Make the adapters available to applications by adding libraries to appropriate STEPLIB concatenations. To cater for SNAP dumps issued by an adapter, allocate a CSQSNAP DDname. Consider using CSQBDEFV to improve the portability of your application programs

- *Repeat this task for each IBM MQ queue manager as required.*
- *You might need to perform this task when migrating from a previous version.*

To make the adapters available to batch and other applications using batch connections, add the following IBM MQ libraries to the STEPLIB concatenation for your batch application :

- thlqual.SCSQANL *x*
- thlqual.SCSQAUTH

where *x* is the language letter for your national language. (You do not need to do this if the libraries are in the LPA or the link list.)

For TSO applications add the libraries to the STEPLIB concatenation in the TSO logon procedure or activate them using the TSO command TSOLIB.

If the adapter detects an unexpected IBM MQ error, it issues an z/OS SNAP dump to DDname CSQSNAP, and issues reason code `MQRC_UNEXPECTED_ERROR` to the application. If the CSQSNAP DD statement is not in the application JCL or CSQSNAP is not allocated to a data set under TSO, no dump is taken. If this happens, you could include the CSQSNAP DD statement in the application JCL or allocate CSQSNAP to a data set under TSO and rerun the application. However, because some problems are intermittent, it is recommended that you include a CSQSNAP statement in the application JCL or allocate CSQSNAP to a data set in the TSO logon procedure to capture the reason for failure at the time it occurs.

The supplied program CSQBDEFV improves the portability of your application programs. In CSQBDEFV, you can specify the name of a queue manager, or queue sharing group, to be connected to rather than specifying it in the `MQCONN` or `MQCONNX` call in an application program. You can create a new version of CSQBDEFV for each queue manager, or queue sharing group. To do this, follow these steps:

1. Copy the IBM MQ assembler program CSQBDEFV from thlqual.SCSQASMS to a user library.
2. The supplied program contains the default subsystem name CSQ1. You can retain this name for testing and installation verification. For production subsystems, you can change the NAME=CSQ1 to your one- to four-character subsystem name, or use CSQ1.

   If you are using queue-sharing groups, you can specify a queue-sharing group name instead of CSQ1. If you do this, the program issues a connect request to an active queue manager within that group.
3. Assemble and link-edit the program to produce the CSQBDEFV load module. For the assembly, include the library thlqual.SCSQMACS in your SYSLIB concatenation; use the link-edit parameters `RENT,AMODE=31,RMODE=ANY`. This is shown in the sample JCL in thlqual.SCSQPROC(CSQ4DEFV). Then include the load library in the z/OS Batch or the TSO STEPLIB, ahead of thlqual.SCSQAUTH.

Related concepts

"Task 20: Set up the operations and control panels" on page 572
To set up the operations and control panels you must first set up the libraries that contain the required panels, EXECs, messages, and tables. To do this, you must take into account which national language feature is to be used for the panels. When you have done this, you can optionally update the main ISPF menu for IBM MQ operations and control panels and change the function key settings.

Task 20: Set up the operations and control panels

To set up the operations and control panels you must first set up the libraries that contain the required panels, EXECs, messages, and tables. To do this, you must take into account which national language feature is to be used for the panels. When you have done this, you can optionally update the main ISPF menu for IBM MQ operations and control panels and change the function key settings.

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*

- *You might need to perform this task when migrating from a previous version.*

Setting up the libraries

Follow these steps to set up the IBM MQ operations and control panels:

1. Ensure that all the libraries contained in your concatenations are either in the same format (F, FB, V, VB) and have the same block size, or are in order of decreasing block sizes. Otherwise, you might have problems trying to use these panels.
2. Include the library thlqual.SCSQEXEC in your SYSEXEC or SYSPROC concatenation or activate it using the TSO ALTLIB command. This library, which is allocated with a fixed-block 80 record format during installation, contains the required EXECs.

   It is preferable to put the library into your SYSEXEC concatenation. However, if you want to put it in SYSPROC, the library must have a record length of 80 bytes.
3. Add thlqual.SCSQAUTH and thlqual.SCSQANLx to the TSO logon procedure STEPLIB or activate it using the TSO TSOLIB command, if it is not in the link list or the LPA.
4. You can either add the IBM MQ panel libraries permanently to your ISPF library setup, or allow them to be set up dynamically when the panels are used. For the former choice, you need to do the following:
   a. Include the library containing the operations and control panel definitions in your ISPPLIB concatenation. The name is thlqual.SCSQPNLx, where x is the language letter for your national language.
   b. Include the library containing the required tables in your ISPTLIB concatenation. The name is thlqual.SCSQTBLx, where x is the language letter for your national language.
   c. Include the library containing the required messages in your ISPMLIB concatenation. The name is thlqual.SCSQMSGx, where x is the language letter for your national language.
   d. Include the library containing the required load modules in your ISPLLIB concatenation. The name of this library is thlqual.SCSQAUTH.
5. Test that you can access the IBM MQ panels from the TSO Command Processor panel. This is usually option 6 on the ISPF/PDF Primary Options Menu. The name of the EXEC that you run is CSQOREXX. There are no parameters to specify if you have put the IBM MQ libraries permanently in your ISPF setup as in step 4. If you have not, use the following:

   ```
   CSQOREXX thlqual langletter
   ```

   where `langletter` is a letter identifying the national language to be used:

   C
   Simplified Chinese

   E
   U.S. English (mixed case)

   F
   French

   K
   Japanese

   U
   U.S. English (uppercase)

Updating the ISPF menu

You can update the ISPF main menu to allow access to the IBM MQ operations and control panels from ISPF. The required setting for &ZSEL is:

```
CMD(%CSQOREXX thlqual langletter)
```

For information about `thlqual` and `langletter`, see Step “5” on page 573.

For more details, see the *z/OS: ISPF Dialog Developer's Guide and Reference* manual.

Updating the function keys and command settings

You can use the normal ISPF procedures for changing the function keys and command settings used by the panels. The application identifier is CSQO.

However, this is *not* recommended because the help information is not updated to reflect any changes that you have made.

Related concepts

“Task 21: Include the IBM MQ dump formatting member” on page 574
To be able to format IBM MQ dumps using the Interactive Problem Control System (IPCS), you must update some system libraries.

Task 21: Include the IBM MQ dump formatting member

To be able to format IBM MQ dumps using the Interactive Problem Control System (IPCS), you must update some system libraries.

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *You need to perform this task when migrating from a previous version.*

To be able to format IBM MQ dumps using the Interactive Problem Control System (IPCS), copy the data set thlqual.SCSQPROC(CSQ7IPCS) to SYS1.PARMLIB. You should not need to edit this data set.

If you have customized the TSO procedure for IPCS, thlqual.SCSQPROC(CSQ7IPCS) can be copied into any library in the IPCSPARM definition. See the *MVS IPCS Customization* manual for details on IPCSPARM.

You must also include the library thlqual.SCSQPNLA in your ISPPLIB concatenation.

To make the dump formatting programs available to your TSO session or IPCS job, you must also include the library thlqual.SCSQAUTH in your STEPLIB concatenation or activate it using the TSO TSOLIB command (even if it is already in the link list or LPA).

Related concepts

“Task 22: Suppress information messages” on page 574
Your IBM MQ system might produce a large number of information messages. You can prevent selected messages being sent to the console or to the hardcopy log.

Task 22: Suppress information messages

Your IBM MQ system might produce a large number of information messages. You can prevent selected messages being sent to the console or to the hardcopy log.

- *You need to perform this task once for each z/OS system where you want to run IBM MQ.*
- *You do not need to perform this task when migrating from a previous version.*

If your IBM MQ system is heavily used, with many channels stopping and starting, a large number of information messages are sent to the z/OS console and hardcopy log. The IBM MQ - IMS bridge and buffer manager might also produce a large number of information messages.

If required, you can suppress some of these console messages by using the z/OS message processing facility list, specified by the MPFLSTxx members of SYS1.PARMLIB. The messages you specify still appear on the hardcopy log, but not on the console.

Sample thlqual.SCSQPROC(CSQ4MPFL) shows suggested settings for MPFLSTxx. See the *MVS Initialization and Tuning Reference* manual for more information about MPFLSTxx.

If you want to suppress selected information messages on the hardcopy log, you can use the z/OS installation exit IEAVMXIT. You can set the following bit switches ON for the required messages:

CTXTRDTM
Delete the message.

The message is not displayed on consoles or logged in hardcopy.

CTXTESJL
Suppress from job log.

The message does not go into the JES job log.

CTXTNWTP
Do not carry out WTP processing.

The message is not sent to a TSO terminal or to the system message data set of a batch job.

Note:

1. For full details on the other parameters, refer to the MVS Installation Exits documentation.
2. You are not recommended to suppress messages other than those in the suggested suppression list, CSQ4MPFL.

In addition you can specify the extra parameter:

EXCLMSG
Specifies a list of messages to be excluded from any log.

Messages in this list are not sent to the z/OS console and hardcopy log. See EXCLMSG in "Using CSQ6SYSP" on page 550 for further information.

Related concepts
"Testing your queue manager on z/OS" on page 579
When you have customized or migrated your queue manager, you can test it by running some of the sample applications shipped with IBM MQ.

Task 23: Create procedures for Advanced Message Security

Each IBM MQ subsystem that is to be configured to use Advanced Message Security requires a cataloged procedure to start the AMS address space. You can create your own or use the IBM-supplied procedure library.

For each IBM MQ subsystem that is to be configured to use Advanced Message Security tailor a copy of sample procedure CSQ4AMSM. To do this, perform the following steps:

1. Copy the sample started task procedure *thlqual*.SCSQPROC(CSQ4AMSM) to your SYS1.PROCLIB or, if you are not using SYS1.PROCLIB, your procedure library. Name the procedure xxxxAMSM, where xxxx is the name of your IBM MQ subsystem. For example, CSQ1AMSM would be the AMS started task procedure for queue manager CSQ1.
2. Make a copy for each IBM MQ subsystem that you are going to use.
3. Tailor the procedures to your requirements using the instructions in the sample procedure CSQ4AMSM. You can also use symbolic parameters in the JCL to allow the procedure to be modified when it is started.
4. Review and optionally change the parameters passed to the AMS task using the Language Environment® _CEE_ENVFILE file. The sample thlqual.SCSQPROC(CSQ40ENV) lists the supported parameters.

Note: This task should be repeated for each IBM MQ queue manager.

Task 24: Set up the started task user Advanced Message Security

The IBM MQ Advanced Message Security task requires a user ID that allows it to be known as a UNIX System Services process.

In addition, the users that the task works on behalf of must also have an appropriate definition of a UNIX UID (user ID) and GID (group ID) so these users are known as UNIX System Services users. For more information on defining UNIX System Services UIDs and GIDs, see *z/OS: Security Server RACF Security Administrator's Guide*.

z/OS: UNIX System Services Planning compares traditional UNIX security to z/OS security. The primary difference between traditional UNIX security and z/OS security is that the Kernel services support two levels of appropriate privileges: UNIX level and z/OS UNIX level.

Depending on your installation's security policy, the IBM MQ Advanced Message Security task can either run with superuser authority (uid(0)), or with its RACF identity permitted to the RACF FACILITY class BPX.DAEMON and BPX.SERVER profiles, as this task must be able to assume the RACF identity of its users.

If the latter method is used, or you have already activated the BPX.DAEMON or BPX.SERVER profiles, the IBM MQ Advanced Message Security task program (thlqual.SCSQAUTH(CSQ0DSRV)) must be located in RACF program-controlled libraries.

Review *z/OS: UNIX System Services Planning* to ensure that you understand the security differences between traditional UNIX security and z/OS UNIX security. This allows you to administer the IBM MQ Advanced Message Security task according to your installation's security policy for deploying and running privileged UNIX System Services processes.

For reference, the publications useful to this review are:

- *z/OS: UNIX System Services Planning*.
- *z/OS: Security Server RACF Security Administrator's Guide*.

Note: Choose the user ID for this task carefully because the IBM MQ Advanced Message Security recipient certificates are loaded into a key ring associated with this user ID. This consideration is discussed in Using certificates on z/OS .

The steps shown here describe how to set up the IBM MQ Advanced Message Security started task user. The steps use RACF commands as examples. If you are using a different security manager, you should use equivalent commands.

Note: The examples in this section assume that you have activated generic profile command processing for the RACF STARTED, FACILITY, and SURROGAT classes and generic profile checking. For more information on how RACF handles generic profiles, see *z/OS: Security Server RACF Command Language Reference*.

1. First define RACF user profiles for the IBM MQ Advanced Message Security started task user. These can be the same user.

   ```
   ADDUSER WMQAMSM NAME('IBM MQ Advanced Message Security user') OMVS (UID(0)) DFLTGRP(group)
   ```

   Select a default 'group' as appropriate to your installation standards.

   Note: If you do not want to grant USS superuser authority (UID(0)), then you must permit the IBM MQ Advanced Message Security user ID to the BPX.DAEMON and BPX.SERVER facility class profiles:

   ```
   PERMIT BPX.DAEMON CLASS(FACILITY) ID(WMQAMSM) ACCESS(READ)
   ```

   and the IBM MQ Advanced Message Security task program (*thlqual*.SCSQAUTH(CSQ0DSRV)) must be located in a RACF program-controlled library.

   To make your SCSQAUTH library program controlled, you can use the following command:

   ```
   RALTER PROGRAM * ADDMEM('thlqual.SCSQAUTH'//NOPADCHK) -or-
   ```

```
RALTER PROGRAM ** ADDMEM('thlqual.SCSQAUTH'//NOPADCHK)
SETROPTS WHEN(PROGRAM) REFRESH
```

You must also enable program control for the national language library (*thlqual*.SCSQANLx) that is used by the IBM MQ Advanced Message Security task.

2. Determine if the RACF STARTED class is active. If it is not, activate the RACF STARTED class:

```
SETROPTS CLASSACT(STARTED)
```

3. Define a started class profile for the IBM MQ Advanced Message Security tasks, specifying the user IDs you selected or created in step 1:

```
RDEFINE STARTED qmgr AMSM.* STDATA(USER(WMQAMSM))
```

where *qmgr* is the name of prefix of the started task name. For example, the started tasks may be named CSQ1AMSM. In this case, you would substitute `qmgr AMSM.*` with `CSQ1AMSM.*`.

The started task names must be named `qmgr AMSM.*`.

4. Use the SETROPTS RACF command to refresh the in-storage RACLISTed started class profiles:

```
SETROPTS RACLIST(STARTED) REFRESH
```

5. The IBM MQ Advanced Message Security task temporarily assumes the identity of the host user ID of the client requestor during protection processing of IBM MQ messages. Therefore, it is necessary to define profiles in the SURROGAT class for each user ID that can make requests.

This can be done with a single generic profile if the RACF SURROGAT class is active. The check is ignored if the SURROGAT class is not active. The SURROGAT profiles needed are described in *z/OS: UNIX System Services Planning*.

To define profiles in the SURROGAT class:

a. Activate the RACF SURROGAT class using the RACF SETROPTS command:

```
SETROPTS CLASSACT(SURROGAT)
```

b. Activate generic profile processing for the RACF SURROGAT class:

```
SETROPTS GENERIC(SURROGAT)
```

c. Activate generic profile command processing for the RACF SURROGAT class:

```
SETROPTS GENCMD(SURROGAT)
```

d. Define a surrogate class generic profile:

```
RDEFINE SURROGAT BPX.SRV.* UACC(NONE)
```

e. Permit the IBM MQ Advanced Message Security user ID to the generic SURROGAT class profile:

```
PERMIT BPX.SRV.* CLASS(SURROGAT) ID(WMQAMSM) ACCESS(UPDATE)
```

Note: You can define more specific profiles if you want to restrict specific users to be processed by the IBM MQ Advanced Message Security task, as described in *z/OS: UNIX System Services Planning*.

f. Permit the IBM MQ Advanced Message Security user ID to the BPX.SERVER facility (if not already done in Creating the certificates and key rings):

```
PERMIT BPX.SERVER CLASS(FACILITY) ID(WMQAMSM) ACCESS(READ)
```

6. The IBM MQ Advanced Message Security task uses the facilities provided by z/OS System SSL services to open SAF-managed key rings. The underlying System Authorization Facility (SAF) that accesses the contents of the key rings is controlled by RACF, or an equivalent security manager.

   This service is the IRRSDL00 (R_datalib) callable service. This callable service is protected with the same profiles used to protect the RACF RACDCERT commands that are defined to the RACF FACILITY class. Thus, the IBM MQ Advanced Message Security user ID must be permitted to the profiles using these commands:

   a. If you have not already done so, define a RACF generic profile to the RACF FACILITY class that protects the RACDCERT command and the IRRSDL00 callable service:

   ```
   RDEFINE FACILITY IRR.DIGTCERT.* UACC(NONE)
   SETROPTS RACLIST(FACILITY) REFRESH
   ```

   b. Grant authority to the started task user ID to the RACF generic profile:

   ```
   PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(WMQAMSM) ACC(READ)
   ```

   Alternatively, you can grant READ access to the data service task user's keyring in the RDATALIB class as follows:

   ```
   PERMIT WMQASMD.DRQ.AMS.KEYRING.LST CLASS(RDATALIB) ID(WMQAMSM) ACC(READ)
   ```

Resource security for AMS

The started task user requires read authority to the SYSTEM.PROTECTION.POLICY.QUEUE.

The started task user requires authority to connect to the queue manager as a BATCH application. For further information, see Connection security profiles for batch connections.

Task 25: Grant RACDCERT permissions to the security administrator for Advanced Message Security

Your IBM MQ Advanced Message Security security administrator requires authority to use the RACDCERT command to create and manage digital certificates.

Identify the appropriate user ID for this role and grant permission to use the RACDCERT command. For example:

```
PERMIT IRR.DIGTCERT.* CLASS(FACILITY) ID(admin) ACCESS(CONTROL)
SETROPTS RACLIST(FACILITY) REFRESH
```

where `admin` is the user ID of your IBM MQ Advanced Message Security security administrator.

Task 26: Grant users resource permissions for IBM MQ Advanced Message Security

IBM MQ Advanced Message Security users require relevant resource permissions.

IBM MQ Advanced Message Security users, that is users that are putting or getting IBM MQ Advanced Message Security protected messages, require:

- An OMVS segment associated with their user id

- Permissions for IRR.DIGTCERT.LISTRING or RDATALIB
- Permissions for ICSF class CSFSERV and CSFKEYS profiles

The IBM MQ Advanced Message Security task temporarily assumes the identity of its clients; that is, the task acts as a surrogate of the z/OS user ID of users of IBM MQ Advanced Message Security during the processing of IBM MQ messages to queues that are protected by IBM MQ Advanced Message Security.

In order for the task to assume the z/OS identity of a user, the client z/OS user ID must have a defined OMVS segment associated with its user profile.

As an administration aid, RACF provides the ability to define a default OMVS segment that may be associated with RACF user and group profiles. This default is used if the z/OS user ID or group profile does not have an OMVS segment explicitly defined. If you plan to have a large number of users using IBM MQ Advanced Message Security, you may choose to use this default rather than explicitly defining the OMVS segment for each user.

The *z/OS: Security Server RACF Security Administrator's Guide* contains the detailed procedure for defining default OMVS segments. Review the procedure as outlined in this publication to determine if the definition of default OMVS segments in RACF User and Group profiles is appropriate to your installation.

To grant READ permission to the IRR.DIGTCERT.LISTRING class facility to all IBM MQ Advanced Message Security users, issue this command:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(READ)
```

or grant READ permission on a per user basis by issuing this command:

```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(userid) ACCESS(READ)
```

where userid is the name of the IBM MQ Advanced Message Security user.

Alternatively, you can use the RDATALIB class to grant access to specific keyrings (the RDATALIB permissions take precedence over IRR.DIGTCERT.LISTRING permissions). For example:

```
PERMIT user.DRQ.AMS.KEYRING.LST CLASS(RDATALIB) ID(user) ACC(READ)
```

If you are using ICSF-managed certificates and private keys, IBM MQ Advanced Message Security users require access to certain class CSFSERV and CSFKEYS profiles. This access is detailed in the following table:

Table 46. Required user access to class CSFSERV and CSFKEYS profiles

Class	Profile	Permission
CSFSERV	CSFDSG	READ
CSFSERV	CSFPKE	READ
CSFSERV	CSFPKD	READ
CSFSERV	CSFDSV	READ
CSFKEYS	ICSF PKDS Label	READ

Testing your queue manager on z/OS

When you have customized or migrated your queue manager, you can test it by running some of the sample applications shipped with IBM MQ.

You can compile and link-edit whichever of the other samples are appropriate to your installation using the sample JCL supplied.

See the following links for instructions on how to test your queue manager on z/OS:

- “Running the basic installation verification program” on page 580
- “Testing for queue-sharing groups” on page 583
- “Testing for distributed queuing” on page 585
- “Testing for C, C++, COBOL, PL/I, and CICS” on page 587

Related concepts

IBM MQ for z/OS concepts

“Configuring queue managers on z/OS” on page 522

Use these instructions to configure queue managers on IBM MQ for z/OS.

Related tasks

Planning your IBM MQ environment on z/OS

Administering IBM MQ for z/OS

Running the basic installation verification program

After you have installed and customized IBM MQ, you can use the supplied installation verification program, CSQ4IVP1, to confirm that IBM MQ is operational.

The basic installation verification program is a batch assembler IVP that verifies the base IBM MQ without using the C, COBOL, or CICS samples.

The Batch Assembler IVP is link-edited by SMP/E and the load modules are shipped in library thlqual.SCSQLOAD.

After you have completed both the SMP/E APPLY step and the customization steps, run the Batch Assembler IVP.

See these sections for further details:

- Overview of the CSQ4IVP1 application
- Preparing to run CSQ4IVP1
- Running CSQ4IVP1
- Checking the results of CSQ4IVP1

Overview of the CSQ4IVP1 application

CSQ4IVP1 is a batch application that connects to your IBM MQ subsystem and performs these basic functions:

- Issues IBM MQ calls
- Communicates with the command server
- Verifies that triggering is active
- Generates and deletes a dynamic queue
- Verifies message expiry processing
- Verifies message commit processing

Preparing to run CSQ4IVP1

Before you run CSQ4IVP1:

1. Check that the IVP entries are in the CSQINP2 data set concatenation in the queue manager startup program. The IVP entries are supplied in member thlqual.SCSQPROC(CSQ4IVPQ). If not, add the definitions supplied in thlqual.SCSQPROC(CSQ4IVPQ) to your CSQINP2 concatenation. If the queue manager is currently running, you need to restart it so that these definitions can take effect.

2. The sample JCL, CSQ4IVPR, required to run the installation verification program is in library thlqual.SCSQPROC.

   Customize the CSQ4IVPR JCL with the high-level qualifier for the IBM MQ libraries, the national language you want to use, the four-character IBM MQ queue manager name, and the destination for the job output.
3. Update RACF to allow CSQ4IVP1 to access its resources if IBM MQ security is active.

   To run CSQ4IVP1 when IBM MQ security is enabled, you need a RACF user ID with authority to access the objects. For details of defining resources to RACF, see Setting up security on z/OS . The user ID that runs the IVP must have the following access authority:

Authority	**Profile**	**Class**
READ	ssid.DISPLAY.PROCESS	MQCMDS
UPDATE	ssid.SYSTEM.COMMAND.INPUT	MQQUEUE
UPDATE	ssid.SYSTEM.COMMAND.REPLY.MODEL	MQQUEUE
UPDATE	ssid.CSQ4IVP1.**	MQQUEUE
READ	ssid.BATCH	MQCONN

   These requirements assume that all IBM MQ security is active. The RACF commands to activate IBM MQ security are shown in Figure 91 on page 581. This example assumes that the queue manager name is CSQ1 and that the user ID of the person running sample CSQ4IVP1 is TS101.

```
RDEFINE MQCMDS CSQ1.DISPLAY.PROCESS
PERMIT CSQ1.DISPLAY.PROCESS CLASS(MQCMDS) ID(TS101) ACCESS(READ)

RDEFINE MQQUEUE CSQ1.SYSTEM.COMMAND.INPUT
PERMIT CSQ1.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.COMMAND.REPLY.MODEL
PERMIT CSQ1.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.CSQ4IVP1.**
PERMIT CSQ1.CSQ4IVP1.** CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)

RDEFINE MQCONN CSQ1.BATCH
PERMIT CSQ1.BATCH CLASS(MQCONN) ID(TS101) ACCESS(READ)
```

Figure 91. RACF commands for CSQ4IVP1

Running CSQ4IVP1

When you have completed these steps, start your queue manager. If the queue manager is already running and you have changed CSQINP2, you must stop the queue manager and restart it.

The IVP runs as a batch job. Customize the job card to meet the submission requirements of your installation.

Checking the results of CSQ4IVP1

The IVP is split into 10 stages; each stage must complete with a zero completion code before the next stage is run. The IVP generates a report, listing:

- The name of queue manager that is being connected to.
- A one-line message showing the completion code and the reason code returned from each stage.
- A one-line informational message where appropriate.

A sample report is provided in Figure 92 on page 583

For an explanation of the completion and reason codes, see IBM MQ for z/OS messages, completion, and reason codes.

Some stages have more than one IBM MQ call and, in the event of failure, a message is issued indicating the specific IBM MQ call that returned the failure. Also, for some stages the IVP puts explanatory and diagnostic information into a comment field.

The IVP job requests exclusive control of certain queue manager objects and therefore should be single threaded through the system. However, there is no limit to the number of times the IVP can be run against your queue manager.

The functions performed by each stage are:

Stage 1
Connect to the queue manager by issuing the `MQCONN` API call.

Stage 2
Determine the name of the system-command input queue used by the command server to retrieve request messages. This queue receives display requests from Stage 5.

To do this, the sequence of calls is:

1. Issue an `MQOPEN` call, specifying the queue manager name, to open the queue manager object.
2. Issue an `MQINQ` call to find out the name of the system-command input queue.
3. Issue an `MQINQ` call to find out about various queue manager event switches.
4. Issue an `MQCLOSE` call to close the queue manager object.

On successful completion of this stage, the name of the system-command input queue is displayed in the comment field.

Stage 3
Open an initiation queue using an `MQOPEN` call.

This queue is opened at this stage in anticipation of a trigger message, which arrives as a result of the command server replying to the request from Stage 5. The queue must be opened for input to meet the triggering criteria.

Stage 4
Create a permanent dynamic queue using the CSQ4IVP1.MODEL queue as a model. The dynamic queue has the same attributes as the model from which it was created. This means that when the replies from the command server request in Stage 5 are written to this queue, a trigger message is written to the initiation queue opened in Stage 3.

Upon successful completion of this stage, the name of the permanent dynamic queue is indicated in the comment field.

Stage 5
Issue an `MQPUT1` request to the command server command queue.

A message of type MQMT_REQUEST is written to the system-command input queue requesting a display of process CSQ4IVP1. The message descriptor for the message specifies the permanent dynamic queue created in Stage 4 as the reply-to queue for the command server's response.

Stage 6
Issue an **MQGET** request from the initiation queue. At this stage, a GET WAIT with an interval of 1 minute is issued against the initiation queue opened in Stage 3. The message returned is expected to be the trigger message generated by the command server's response messages being written to the reply-to queue.

Stage 7
Delete the permanent dynamic queue created in Stage 4. As the queue still has messages on it, the MQCO_PURGE_DELETE option is used.

Stage 8

1. Open a dynamic queue.
2. MQPUT a message with an expiry interval set.
3. Wait for the message to expire.
4. Attempt to MQGET the expired message.
5. MQCLOSE the queue.

Stage 9

1. Open a dynamic queue.
2. MQPUT a message.
3. Issue MQCMIT to commit the current unit of work.
4. MQGET the message.
5. Issue MQBACK to backout the message.
6. MQGET the same message and ensure that the backout count is set to 1.
7. Issue MQCLOSE to close the queue.

Stage 10

Disconnect from the queue manager using `MQDISC` .

After running the IVP, you can delete any objects that you no longer require.

If the IVP does not run successfully, try each step manually to find out which function is failing.

```
DATE : 2005.035                  IBM MQ for z/OS - V6            PAGE : 0001
INSTALLATION VERIFICATION PROGRAM
PARAMETERS ACCEPTED. PROGRAM WILL CONNECT TO : CSQ1
,OBJECT QUALIFER : CSQ4IVP1
INSTALLATION VERIFICATION BEGINS :
STAGE 01 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR BRIDGE EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS EXCP FOR CHANNEL EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR SSL EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR INHIBITED EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR LOCAL EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR PERFORMENCE EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR REMOTE EVENTS
STAGE 02 INFO: QMGR EVENT SWITCH IS OFF FOR START/STOP EVENTS
STAGE 02 COMPLETE. COMPCODE : 0000 REASON CODE : 0000 SYSTEM.COMMAND.INPUT
STAGE 03 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
STAGE 04 COMPLETE. COMPCODE : 0000 REASON CODE : 0000 CSQ4IVP1.BAB9810EFEAC8980
STAGE 05 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
STAGE 06 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
STAGE 07 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
STAGE 08 COMPLETE. COMPCODE : 0000 REASON CODE : 0000 CSQ4IVP1.BAB9810F0070E645
STAGE 09 COMPLETE. COMPCODE : 0000 REASON CODE : 0000 CSQ4IVP1.BAB9812BA8706803
STAGE 10 COMPLETE. COMPCODE : 0000 REASON CODE : 0000
>>>>>>>>>>   END OF REPORT   <<<<<<<<<<
```

Figure 92. Sample report from CSQ4IVP1

Testing for queue-sharing groups

The basic installation verification program CSQ4IVP1 tests non-shared queues.

CSQ4IVP1 can be used whether the queue manager is a member of a queue-sharing group or not. After running the basic IVP, you can test for shared queues by using the CSQ4IVP1 installation verification program with different queues. Also this tests that Db2 and the coupling facility are set up correctly.

Preparing to run CSQ4IVP1 for a queue-sharing group

Before you run CSQ4IVP1:

1. Add the coupling facility structure that the IVP uses to your CFRM policy data set, as described in "Task 10: Set up the coupling facility" on page 541. The supplied samples use a structure called APPLICATION1, but you can change this if you want.
2. Check that the IVP entries are in the CSQINP2 data set concatenation in the queue manager startup program. The IVP entries are supplied in member thlqual.SCSQPROC(CSQ4IVPG). If they are not, add the definitions supplied in thlqual.SCSQPROC(CSQ4IVPG) to your CSQINP2 concatenation. If the queue manager is currently running, you need to restart it so that these definitions can take effect.
3. Change the name of the coupling facility structure used in thlqual.SCSQPROC(CSQ4IVPG) if necessary.
4. The sample JCL, CSQ4IVPS, required to run the installation verification program for a queue-sharing group is in library thlqual.SCSQPROC.

   Customize the CSQ4IVPS JCL with the high-level qualifier for the IBM MQ libraries, the national language you want to use, the four-character IBM MQ queue manager name, and the destination for the job output.
5. Update RACF to allow CSQ4IVP1 to access its resources if IBM MQ security is active.

   To run CSQ4IVP1 when IBM MQ security is enabled, you need a RACF user ID with authority to access the objects. For details of defining resources to RACF, see Setting up security on z/OS . The user ID that runs the IVP must have the following access authority in addition to that required to run the basic IVP:

Authority	Profile	Class
UPDATE	ssid.CSQ4IVPG.**	MQQUEUE

   These requirements assume that all IBM MQ security is active. The RACF commands to activate IBM MQ security are shown in Figure 93 on page 584. This example assumes that the queue manager name is CSQ1 and that the user ID of the person running sample CSQ4IVP1 is TS101.

```
RDEFINE MQQUEUE CSQ1.CSQ4IVPG.**
PERMIT CSQ1.CSQ4IVPG.** CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)
```

Figure 93. RACF commands for CSQ4IVP1 for a queue-sharing group

Running CSQ4IVP1 for a queue-sharing group

When you have completed these steps, start your queue manager. If the queue manager is already running and you have changed CSQINP2, you must stop the queue manager and restart it.

The IVP runs as a batch job. Customize the job card to meet the submission requirements of your installation.

Checking the results of CSQ4IVP1 for a queue-sharing group

The IVP for queue-sharing groups works in the same way as the basic IVP, except that the queues that are created are called CSQIVPG. *xx*. Follow the instructions given in "Checking the results of CSQ4IVP1" on page 581 to check the results of the IVP for queue-sharing groups.

Testing for distributed queuing

You can use the supplied installation verification program, CSQ4IVPX, to confirm that distributed queuing is operational.

Overview of CSQ4IVPX job

CSQ4IVPX is a batch job that starts the channel initiator and issues the IBM MQ DISPLAY CHINIT command. This verifies that all major aspects of distributed queuing are operational, while avoiding the need to set up channel and network definitions.

Preparing to run CSQ4IVPX

Before you run CSQ4IVPX:

1. The sample JCL, CSQ4IVPX, required to run the installation verification program is in library thlqual.SCSQPROC.

   Customize the CSQ4IVPX JCL with the high-level qualifier for the IBM MQ libraries, the national language you want to use, the four-character queue manager name, and the destination for the job output.
2. Update RACF to allow CSQ4IVPX to access its resources if IBM MQ security is active. To run CSQ4IVPX when IBM MQ security is enabled, you need a RACF user ID with authority to access the objects. For details of defining resources to RACF, see Setting up security on z/OS . The user ID that runs the IVP must have the following access authority:

Authority	Profile	Class
CONTROL	ssid.START.CHINIT and ssid.STOP.CHINIT	MQCMDS
UPDATE	ssid.SYSTEM.COMMAND.INPUT	MQQUEUE
UPDATE	ssid.SYSTEM.CSQUTIL.*	MQQUEUE
READ	ssid.BATCH	MQCONN
READ	ssid.DISPLAY.CHINIT	MQCMDS

   These requirements assume that the connection security profile ssid.CHIN has been defined (as shown in Connection security profiles for the channel initiator), and that all IBM MQ security is active. The RACF commands to do this are shown in Figure 94 on page 586. This example assumes that:

   - The queue manager name is CSQ1
   - The user ID of the person running sample CSQ4IVPX is TS101
   - The channel initiator address space is running under the user ID CSQ1MSTR
3. Update RACF to allow the channel initiator address space the following access authority:

Authority	Profile	Class
READ	ssid.CHIN	MQCONN
UPDATE	ssid.SYSTEM.COMMAND.INPUT	MQQUEUE
UPDATE	ssid.SYSTEM.CHANNEL.INITQ	MQQUEUE
UPDATE	ssid.SYSTEM.CHANNEL.SYNCQ	MQQUEUE
ALTER	ssid.SYSTEM.CLUSTER.COMMAND.QUEUE	MQQUEUE
UPDATE	ssid.SYSTEM.CLUSTER.TRANSMIT.QUEUE	MQQUEUE
ALTER	ssid.SYSTEM.CLUSTER.REPOSITORY.QUEUE	MQQUEUE
CONTROL	ssid.CONTEXT.**	MQADMIN

The RACF commands to do this are also shown in Figure 94 on page 586.

```
RDEFINE MQCMDS CSQ1.DISPLAY.DQM
PERMIT CSQ1.DISPLAY.DQM CLASS(MQCMDS) ID(TS101) ACCESS(READ)

RDEFINE MQCMDS CSQ1.START.CHINIT
PERMIT CSQ1.START.CHINIT CLASS(MQCMDS) ID(TS101) ACCESS(CONTROL)

RDEFINE MQCMDS CSQ1.STOP.CHINIT
PERMIT CSQ1.STOP.CHINIT CLASS(MQCMDS) ID(TS101) ACCESS(CONTROL)

RDEFINE MQQUEUE CSQ1.SYSTEM.COMMAND.INPUT
PERMIT CSQ1.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) ID(TS101,CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.CSQUTIL.*
PERMIT CSQ1.SYSTEM.CSQUTIL.* CLASS(MQQUEUE) ID(TS101) ACCESS(UPDATE)

RDEFINE MQCONN CSQ1.BATCH
PERMIT CSQ1.BATCH CLASS(MQCONN) ID(TS101) ACCESS(READ)

RDEFINE MQCONN CSQ1.CHIN
PERMIT CSQ1.CHIN CLASS(MQCONN) ID(CSQ1MSTR) ACCESS(READ)

RDEFINE MQQUEUE CSQ1.SYSTEM.CHANNEL.SYNCQ
PERMIT CSQ1.SYSTEM.CHANNEL.SYNCQ CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.CLUSTER.COMMAND.QUEUE
PERMIT CSQ1.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(ALTER)

RDEFINE MQQUEUE CSQ1.SYSTEM.CLUSTER.TRANSMIT.QUEUE
PERMIT CSQ1.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQQUEUE CSQ1.SYSTEM.CLUSTER.REPOSITORY.QUEUE
PERMIT CSQ1.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(ALTER)

RDEFINE MQQUEUE CSQ1.SYSTEM.CHANNEL.INITQ
PERMIT CSQ1.SYSTEM.CHANNEL.INITQ CLASS(MQQUEUE) ID(CSQ1MSTR) ACCESS(UPDATE)

RDEFINE MQADMIN CSQ1.CONTEXT.**
PERMIT CSQ1.CONTEXT.** CLASS(MQADMIN) ID(CSQ1MSTR) ACCESS(CONTROL)
```

Figure 94. RACF commands for CSQ4IVPX

Running CSQ4IVPX

When you have completed these steps, start your queue manager.

The IVP runs as a batch job. Customize the job card to meet the submission requirements of your installation.

Checking the results of CSQ4IVPX

CSQ4IVPX runs the CSQUTIL IBM MQ utility to issue three MQSC commands. The SYSPRINT output data set should look like Figure 95 on page 587, although details might differ depending on your queue manager attributes.

- You should see the commands **(1)** each followed by several messages.
- The last message from each command should be "CSQ9022I ... NORMAL COMPLETION" **(2)**.
- The job as a whole should complete with return code zero **(3)**.

```
CSQU000I CSQUTIL IBM MQ for z/OS - V6
CSQU001I CSQUTIL Queue Manager Utility - 2005-05-09 09:06:48
COMMAND
CSQU127I CSQUTIL Executing COMMAND using input from CSQUCMD data set
CSQU120I CSQUTIL Connecting to queue manager CSQ1
CSQU121I CSQUTIL Connected to queue manager CSQ1
CSQU055I CSQUTIL Target queue manager is CSQ1
START CHINIT
(1)
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQM138I +CSQ1 CSQMSCHI CHANNEL INITIATOR STARTING
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQXCRPS ' START CHINIT' NORMAL COMPLETION
(2)
DISPLAY CHINIT
(1)
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQM137I +CSQ1 CSQMDDQM DISPLAY CHINIT COMMAND ACCEPTED
CSQN205I  COUNT=   12, RETURN=00000000, REASON=00000000
CSQX830I +CSQ1 CSQXRDQM Channel initiator active
CSQX002I +CSQ1 CSQXRDQM Queue-sharing group is QSG1
CSQX831I +CSQ1 CSQXRDQM 8 adapter subtasks started, 8 requested
CSQX832I +CSQ1 CSQXRDQM 5 dispatchers started, 5 requested
CSQX833I +CSQ1 CSQXRDQM 0 SSL server subtasks started, 0 requested
CSQX840I +CSQ1 CSQXRDQM 0 channel connections current, maximum 200
CSQX841I +CSQ1 CSQXRDQM 0 channel connections active, maximum 200,
including 0 paused
CSQX842I +CSQ1 CSQXRDQM 0 channel connections starting,
0 stopped, 0 retrying
CSQX836I +CSQ1 Maximum channels - TCP/IP 200, LU 6.2 200
CSQX845I +CSQ1 CSQXRDQM TCP/IP system name is TCPIP
CSQX848I +CSQ1 CSQXRDQM TCP/IP listener INDISP=QMGR not started
CSQX848I +CSQ1 CSQXRDQM TCP/IP listener INDISP=GROUP not started
CSQX849I +CSQ1 CSQXRDQM LU 6.2 listener INDISP=QMGR not started
CSQX849I +CSQ1 CSQXRDQM LU 6.2 listener INDISP=GROUP not started
CSQ9022I +CSQ1 CSQXCRPS ' DISPLAY CHINIT' NORMAL COMPLETION
(2)
STOP CHINIT
(1)
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000004
CSQM137I +CSQ1 CSQMTCHI STOP CHINIT COMMAND ACCEPTED
CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQXCRPS ' STOP CHINIT' NORMAL COMPLETION
(2)
CSQU057I CSQUCMDS 3 commands read
CSQU058I CSQUCMDS 3 commands issued and responses received, 0 failed
CSQU143I CSQUTIL 1 COMMAND statements attempted
CSQU144I CSQUTIL 1 COMMAND statements executed successfully
CSQU148I CSQUTIL Utility completed, return code=0
(3)
```

Figure 95. Example output from CSQ4IVPX

Testing for C, C++, COBOL, PL/I, and CICS

You can test for C, C++, COBOL, PL/I, or CICS, using the sample applications supplied with IBM MQ.

The IVP (CSQ4IVP1) is supplied as a load module, and provides the samples as source modules. You can use these source modules to test different programming language environments.

For more information about sample applications, see Sample programs for IBM MQ for z/OS .

Setting up communications with other queue managers

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

To define your distributed-queuing requirements, you need to define the following items:

- Define the channel initiator procedures and data sets
- Define the channel definitions

- Define the queues and other objects
- Define access security

To enable distributed queuing, you must perform the following three tasks:

- Customize the distributed queuing facility and define the IBM MQ objects required as described in Defining system objects and "Preparing to customize your IBM MQ for z/OS queue managers" on page 522.
- Define access security as described in Security considerations for the channel initiator on z/OS .
- Set up your communications as described in "Setting up communication for z/OS" on page 607.

If you are using queue-sharing groups, see Distributed queuing and queue-sharing groups.

See the following sections for additional considerations for using distributed queuing with IBM MQ for z/OS.

Operator messages

Because the channel initiator uses a number of asynchronously operating dispatchers, operator messages might occur on the log out of chronological sequence.

Channel operation commands

Channel operation commands generally involve two stages. When the command syntax has been checked and the existence of the channel verified, a request is sent to the channel initiator. Message CSQM134I or CSQM137I is sent to the command issuer to indicate the completion of the first stage. When the channel initiator has processed the command, further messages indicating its success or otherwise are sent to the command issuer along with message CSQ9022I or CSQ9023I. Any error messages generated could also be sent to the z/OS console.

All cluster commands except DISPLAY CLUSQMGR, however, work asynchronously. Commands that change object attributes update the object and send a request to the channel initiator. Commands for working with clusters are checked for syntax and a request is sent to the channel initiator. In both cases, message CSQM130I is sent to the command issuer indicating that a request has been sent. This message is followed by message CSQ9022I to indicate that the command has completed successfully, in that a request has been sent. It does not indicate that the cluster request has completed successfully. The requests sent to the channel initiator are processed asynchronously, along with cluster requests received from other members of the cluster. In some cases, these requests must be sent to the whole cluster to determine if they are successful or not. Any errors are reported to the z/OS on the system where the channel initiator is running. They are not sent to the command issuer.

Undelivered-message queue

A Dead Letter handler is provided with IBM MQ for z/OS. See The dead-letter queue handler utility (CSQUDLQH) for more information.

Queues in use

MCAs for receiver channels can keep the destination queues open even when messages are not being transmitted. This behavior results in the queues appearing to be 'in use'.

Security changes

If you change security access for a user ID, the change might not take effect immediately. (See one of Security considerations for the channel initiator on z/OS , Profiles for queue security, and "Task 11: Implement your ESM security controls" on page 542 for more information.)

Communications stopped - TCP

If TCP is stopped for some reason and then restarted, the IBM MQ for z/OS TCP listener waiting on a TCP port is stopped.

Automatic channel-reconnect allows the channel initiator to detect that TCP/IP is unavailable and to automatically restart the TCP/IP listener when TCP/IP returns. This automatic restart alleviates the need for operations staff to notice the problem with TCP/IP and manually restart the listener. While the listener is out of action, the channel initiator can also be used to try the listener again at the interval specified by LSTRTMR. These attempts can continue until TCP/IP returns and the listener successfully restarts automatically. For information about LSTRTMR, see ALTER QMGR and Distributed queuing messages (CSQX...).

Communications stopped - LU6.2

If APPC is stopped, the listener is also stopped. Again, in this case, the listener automatically tries again at the LSTRTMR interval so that, if APPC restarts, the listener can restart too.

If the Db2 fails, shared channels that are already running continue to run, but any new channel start requests fail. When the Db2 is restored new requests are able to complete.

z/OS Automatic Restart Management (ARM)

Automatic restart management (ARM) is a z/OS recovery function that can improve the availability of specific batch jobs or started tasks (for example, subsystems). It can therefore result in a faster resumption of productive work.

To use ARM, you must set up your queue managers and channel initiators in a particular way to make them restart automatically. For information, see Using the z/OS Automatic Restart Manager (ARM).

Related concepts

"Configuring distributed queuing" on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

"Customizing IBM MQ for z/OS" on page 526
Use this topic as a step by step guide for customizing your IBM MQ system.

"Monitoring and controlling channels on z/OS" on page 591
Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers.

"Setting up communication for z/OS" on page 607
When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. To succeed, it is necessary for the connection to be defined and available. This section explains how to define a connection.

"Preparing IBM MQ for z/OS for DQM with queue-sharing groups" on page 611
Use the instructions in this section to configure distributed queuing with queue-sharing groups on IBM MQ for z/OS.

"Setting up communication for IBM MQ for z/OS using queue-sharing groups" on page 615
When a distributed-queuing management channel is started, it attempts to use the connection specified in the channel definition. For this attempt to succeed, it is necessary for the connection to be defined and available.

Defining IBM MQ objects

Use one of the IBM MQ command input methods to define IBM MQ objects. Refer to the information within this topic for further details about defining these objects.

Refer to "Monitoring and controlling channels on z/OS" on page 591 for information about defining objects.

Transmission queues and triggering channels

Define the following:

- A local queue with the usage of XMITQ for each sending message channel.
- Remote queue definitions.

  A remote queue object has three distinct uses, depending upon the way the name and content are specified:

  - Remote queue definition
  - Queue manager alias definition
  - Reply-to queue alias definition

  These three ways are shown in Three ways of using the remote queue definition object.

Use the TRIGDATA field on the transmission queue to trigger the specified channel. For example:

```
DEFINE QLOCAL(MYXMITQ) USAGE(XMITQ) TRIGGER +
INITQ(SYSTEM.CHANNEL.INITQ) TRIGDATA(MYCHANNEL)
DEFINE CHL(MYCHANNEL) CHLTYPE(SDR) TRPTYPE(TCP) +
XMITQ(MYXMITQ) CONNAME('9.20.9.30(1555)')
```

The supplied sample CSQ4INYD gives additional examples of the necessary definitions.

z/OS Loss of connectivity to the CF structure where the synchronization queue for shared channels is defined, or similar problems, might temporarily prevent a channel from starting. After problem resolution, if you are using a trigger type of FIRST and the channel fails to start when it is triggered, you must start the channel manually. If you want to automatically start triggered channels after problem resolution, consider setting the queue manager TRIGINT attribute to a value other than the default. Setting the TRIGINT attribute to a value other than the default causes the channel initiator to retry starting the channel periodically while there are messages on the transmission queue.

Synchronization queue

DQM requires a queue for use with sequence numbers and logical units of work identifiers (LUWID). You must ensure that a queue is available with the name SYSTEM.CHANNEL.SYNCQ (see Planning on z/OS). This queue must be available otherwise the channel initiator cannot start.

Make sure that you define this queue using INDXTYPE(MSGID). This attribute improves the speed at which they can be accessed.

Channel command queues

You need to ensure that a channel command queue exists for your system with the name SYSTEM.CHANNEL.INITQ.

If the channel initiator detects a problem with the SYSTEM.CHANNEL.INITQ, it is unable to continue normally until the problem is corrected. The problem could be one of the following:

- The queue is full
- The queue is not enabled for put
- The page set that the queue is on is full
- The channel initiator does not have the correct security authorization to the queue

If the definition of the queue is changed to GET(DISABLED) while the channel initiator is running, the initiator is unable to get messages from the queue, and terminates.

Starting the channel initiator

Triggering is implemented using the channel initiator. On IBM MQ for z/OS, the initiator is started with the MQSC command `START CHINIT`.

Stopping the channel initiator

The channel initiator is stopped automatically when you stop the queue manager. If you need to stop the channel initiator but not the queue manager, you can use the MQSC command `STOP CHINIT`.

Monitoring and controlling channels on z/OS

Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers.

Each z/OS queue manager has a DQM program (the *channel initiator*) for controlling interconnections to remote queue managers using native z/OS facilities.

The implementation of these panels and commands on z/OS is integrated into the operations and control panels and the MQSC commands. No differentiation is made in the organization of these two sets of panels and commands.

You can also enter commands using Programmable Command Format (PCF) commands. See Automating administration tasks for information about using these commands.

The information in this section applies in all cases where the channel initiator is used for distributed queuing. It applies whether you are using queue-sharing groups, or intra-group queuing.

The DQM channel control function

For an overview of the distributed queue management model, see "Message sending and receiving" on page 145.

The channel control function consists of panels, commands and programs, two synchronization queues, channel command queues, and the channel definitions. This topic is a brief description of the components of the channel control function.

- The channel definitions are held as objects in page set zero or in Db2, like other IBM MQ objects in z/OS.
- You use the operations and control panels, MQSC commands, or PCF commands to:
  - Create, copy, display, alter, and delete channel definitions
  - Start and stop channel initiators and listeners
  - Start, stop, and ping channels, reset channel sequence numbers, and resolve in-doubt messages when links cannot be re-established
  - Display status information about channels
  - Display information about DQM

  In particular, you can use the CSQINPX initialization input data set to issue your MQSC commands. This set can be processed every time you start the channel initiator. For more information, see Initialization commands.
- There are two queues (SYSTEM.CHANNEL.SYNCQ and SYSTEM.QSG.CHANNEL.SYNCQ) used for channel re-synchronization purposes. Define these queues with INDXTYPE(MSGID) for performance reasons.
- The channel command queue (SYSTEM.CHANNEL.INITQ) is used to hold commands for channel initiators, channels, and listeners.
- The channel control function program runs in its own address space, separate from the queue manager, and comprises the channel initiator, listeners, MCAs, trigger monitor, and command handler.
- For queue-sharing groups and shared channels, see Shared queues and queue-sharing groups.
- For intra-group queuing, see Intra-group queuing

Managing your channels on z/OS

Use the links in the following table for information about how to manage your channels, channel initiators, and listeners:

Table 47. Channel tasks

Task to be performed	MQSC command
Define a channel	DEFINE CHANNEL
Alter a channel definition	ALTER CHANNEL
Display a channel definition	DISPLAY CHANNEL
Delete a channel definition	DELETE CHANNEL
Start a channel initiator	START CHINIT
Stop a channel initiator	STOP CHINIT
Display channel initiator information	DISPLAY CHINIT
Start a channel listener	START LISTENER
Stop a channel listener	STOP LISTENER
Start a channel	START CHANNEL
Test a channel	PING CHANNEL
Reset message sequence numbers for a channel	RESET CHANNEL
Resolve in-doubt messages on a channel	RESOLVE CHANNEL
Stop a channel	STOP CHANNEL
Display channel status	DISPLAY CHSTATUS
Display cluster channels	DISPLAY CLUSQMGR

Related concepts

"Using the panels and the commands" on page 593
You can use the MQSC commands, the PCF commands, or the operations and control panels to manage DQM.

"Configuring distributed queuing" on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

"Setting up communications with other queue managers" on page 587
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

"Customizing IBM MQ for z/OS" on page 526
Use this topic as a step by step guide for customizing your IBM MQ system.

"Setting up communication for z/OS" on page 607
When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. To succeed, it is necessary for the connection to be defined and available. This section explains how to define a connection.

"Preparing IBM MQ for z/OS for DQM with queue-sharing groups" on page 611
Use the instructions in this section to configure distributed queuing with queue-sharing groups on IBM MQ for z/OS.

"Setting up communication for IBM MQ for z/OS using queue-sharing groups" on page 615

When a distributed-queuing management channel is started, it attempts to use the connection specified in the channel definition. For this attempt to succeed, it is necessary for the connection to be defined and available.

Using the panels and the commands

You can use the MQSC commands, the PCF commands, or the operations and control panels to manage DQM.

For information about the syntax of the MQSC commands, see Script (MQSC) Commands. For information about PCF commands, see Introduction to Programmable Command Formats.

Using the initial panel

For an introduction to invoking the operations and control panels, using the function keys, and getting help, see Administering IBM MQ for z/OS.

Note: To use the operations and control panels, you must have the correct security authorization; see Administering IBM MQ for z/OS and sub topics for more information. Figure 96 on page 593 shows the panel that is displayed when you start a panel session. The text after the panel explains the actions you perform in this panel.

```
IBM MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action . . . . . . . . . . 1   0. List with filter  4. Manage
1. List or Display  5. Perform
2. Define like    6. Start
3. Alter       7. Stop
8. Command
Object type . . . . . . . . CHANNEL    +
Name . . . . . . . . . . . *
Disposition . . . . . . . . A Q=Qmgr, C=Copy, P=Private, G=Group,
S=Shared, A=All

Connect name . . . . . . . MQ25 - local queue manager or group
Target queue manager . . . MQ25
- connected or remote queue manager for command input
Action queue manager . . . MQ25 - command scope in group
Response wait time . . . . 10  5 - 999 seconds

(C) Copyright IBM Corporation 1993, 2024. All rights reserved.

Command ===> ________________________________________________________________
F1=Help   F2=Split   F3=Exit   F4=Prompt  F9=SwapNext F10=Messages
F12=Cancel
```

Figure 96. The operations and controls initial panel

From this panel, you can:

- Select the action you want to perform by typing in the appropriate number in the **Action** field.
- Specify the object type that you want to work with. Press F4 for a list of object types if you are not sure what they are.
- Display a list of objects of the type specified. Type in an asterisk (*) in the **Name** field and press enter to display a list of objects (of the type specified) that have already been defined on this subsystem. You can then select one or more objects to work with in sequence. Figure 97 on page 594 shows a list of channels produced in this way.
- Specify the disposition in the queue-sharing group of the objects you want to work with in the **Disposition** field. The disposition determines where the object is kept and how the object behaves.
- Choose the local queue manager, or queue-sharing group to which you want to connect in the **Connect name** field. If you want the commands to be issued on a remote queue manager, choose either the **Target queue manager** field or the **Action queue manager** field, depending upon whether the remote

queue manager is not or is a member of a queue-sharing group. If the remote queue manager is not a member of a queue-sharing group, choose the **Target queue manager** field. If the remote queue manager is a member of a queue-sharing group, choose the **Action queue manager** field.

- Choose the wait time for responses to be received in the **Response wait time** field.

```
List Channels - MQ25          Row 1 of 8

Type action codes, then press Enter. Press F11 to display connection status.
1=Display  2=Define like  3=Alter  4=Manage  5=Perform
6=Start   7=Stop

Name         Type      Disposition  Status
<> *           CHANNEL     ALL    MQ25
_  SYSTEM.DEF.CLNTCONN  CLNTCONN   QMGR  MQ25
_  SYSTEM.DEF.CLUSRCVR  CLUSRCVR   QMGR  MQ25 INACTIVE
_  SYSTEM.DEF.CLUSSDR  CLUSSDR    QMGR  MQ25 INACTIVE
_  SYSTEM.DEF.RECEIVER  RECEIVER   QMGR  MQ25 INACTIVE
_  SYSTEM.DEF.REQUESTER REQUESTER   QMGR  MQ25 INACTIVE
_  SYSTEM.DEF.SENDER   SENDER    QMGR  MQ25 INACTIVE
_  SYSTEM.DEF.SERVER   SERVER    QMGR  MQ25 INACTIVE
_  SYSTEM.DEF.SVRCONN  SVRCONN    QMGR  MQ25 INACTIVE
******** End of list ********




Command ===> _______________________________________________________________
F1=Help   F2=Split   F3=Exit   F4=Filter  F5=Refresh  F7=Bkwd
F8=Fwd    F9=SwapNext F10=Messages F11=Status  F12=Cancel
```

Figure 97. Listing channels

z/OS *Defining a channel on z/OS*

On z/OS, you can define a channel using MQSC commands or using the operations and control panels.

To define a channel using the MQSC commands, use DEFINE CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	2 (Define like)
Object type	channel type (for example SENDER) or CHANNEL
Name	
Disposition	The location of the new object.

You are presented with some panels to complete with information about the name and attributes you want for the channel you are defining. They are initialized with the default attribute values. Change any you want before pressing enter.

Note: If you entered CHANNEL in the **`object type`** field, you are presented with the Select a Valid Channel Type panel first.

If you want to define a channel with the same attributes as an existing channel, put the name of the channel you want to copy in the **Name** field on the initial panel. The panels are initialized with the attributes of the existing object.

For information about the channel attributes, see Channel attributes

Note:

1. Name all the channels in your network uniquely. As shown in Network diagram showing all channels, including the source and target queue manager names in the channel name is a good way to do this naming.

After you have defined your channel you must secure your channel, see “Securing a channel” on page 596

Altering a channel definition

You can alter a channel definition using MQSC commands or using the operations and control panels.

To alter a channel definition using the MQSC commands, use ALTER CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	3 (Alter)
Object type	channel type (for example SENDER) or CHANNEL
Name	CHANNEL.TO.ALTER
Disposition	The location of the stored object.

You are presented with some panels containing information about the current attributes of the channel. Change any of the unprotected fields that you want by over typing the new value, and then press enter to change the channel definition.

For information about the channel attributes, see Channel attributes.

Displaying a channel definition

You can display a channel definition using MQSC commands or using the operations and control panels.

To display a channel definition using the MQSC commands, use DISPLAY CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	1 (List or Display)
Object type	channel type (for example SENDER) or CHANNEL
Name	CHANNEL.TO.DISPLAY
Disposition	The location of the object.

You are presented with some panels displaying information about the current attributes of the channel.

For information about the channel attributes, see Channel attributes.

Deleting a channel definition

You can delete a channel definition using MQSC commands or using the operations and control panels.

To delete a channel definition using the MQSC commands, use DELETE CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	4 (Manage)
Object type	channel type (for example SENDER) or CHANNEL
Name	CHANNEL.TO.DELETE
Disposition	The location of the object.

You are presented with another panel. Select function type 1 on this panel.

Press enter to delete the channel definition; you are asked to confirm that you want to delete the channel definition by pressing enter again.

Note: The channel initiator has to be running before a channel definition can be deleted (except for client-connection channels).

Displaying information about the channel initiator

You can display information about the channel initiator using MQSC commands or using the operations and control panels.

To display information about the channel initiator using the MQSC commands, use DISPLAY CHINIT.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	1 (Display)
Object type	SYSTEM
Name	Blank

You are presented with another panel. Select function type 1 on this panel.

Note:

1. Displaying distributed queuing information might take some time if you have lots of channels.
2. The channel initiator has to be running before you can display information about distributed queuing.

Securing a channel

You can secure a channel using MQSC commands or using the operations and control panels.

To secure a channel using the MQSC commands, use SET CHLAUTH.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	8

You are presented with an editor within which you can provide an MQSC command, in this case a CHLAUTH command, see Figure 98 on page 596. When you have finished typing in the command, the plus signs (+) are needed. Type PF3 to exit from the editor and submit the command to the command server.

```
****************************** Top of Data ******************************
000001 SET CHLAUTH(SYSTEM.DEF.SVRCONN) +
000002 TYPE(SSLPEERMAP) +
000003 SSLPEER('CN="John Smith"') +
000004 MCAUSER('PUBLIC')
****** **************************** Bottom of Data ****************************

Command ===>                         Scroll ===> PAGE
F1=Help   F3=Exit   F4=LineEdit F12=Cancel
```

Figure 98. Command Entry

The output of the command is then presented to you, see Figure 99 on page 597

```
****** ***************************** Top of Data ******************************
000001 CSQU000I CSQUTIL IBM MQ for z/OS V7.1.0
000002 CSQU001I CSQUTIL Queue Manager Utility - 2011-04-20 14:42:58
000003 COMMAND TGTQMGR(MQ23) RESPTIME(30)
000004 CSQU127I Executing COMMAND using input from CSQUCMD data set
000005 CSQU120I Connecting to MQ23
000006 CSQU121I Connected to queue manager MQ23
000007 CSQU055I Target queue manager is MQ23
000008 SET CHLAUTH(SYSTEM.DEF.SVRCONN) +
000009 TYPE(SSLPEERMAP) +
000010 SSLPEER('CN="John Smith"') +
000011 MCAUSER('PUBLIC')
000012 CSQN205I  COUNT=    2, RETURN=00000000, REASON=00000000
000013 CSQ9022I !MQ23 CSQMSCA ' SET CHLAUTH' NORMAL COMPLETION
000014 CSQU057I 1 commands read
000015 CSQU058I 1 commands issued and responses received, 0 failed
000016 CSQU143I 1 COMMAND statements attempted
000017 CSQU144I 1 COMMAND statements executed successfully
000018 CSQU148I CSQUTIL Utility completed, return code=0
Command ===>                         Scroll ===> PAGE
F1=Help   F3=Exit   F5=Rfind   F6=Rchange  F9=SwapNext F12=Cancel
```

Figure 99. Command Output

Starting a channel initiator

You can start a channel initiator using MQSC commands or using the operations and control panels.

To start a channel initiator using the MQSC commands, use START CHINIT.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	6 (Start)
Object type	SYSTEM
Name	Blank

The Start a System Function panel is displayed. The text following the following panel explains what action to take:

```
Start a System Function

Select function type, complete fields, then press Enter to start system
function.

Function type . . . . . . . . _   1. Channel initiator
2. Channel listener
Action queue manager . . . : MQ25

Channel initiator
JCL substitution . . . . . ________________________________________________
________________________________________________

Channel listener
Inbound disposition . . . Q G=Group, Q=Qmgr
Transport type . . . . . . _ L=LU6.2, T=TCP/IP
LU name (LU6.2) . . . . . _________________
Port number (TCP/IP) . . . 1414
IP address (TCP/IP) . . . ________________________________________________



Command ===> ________________________________________________________________
F1=Help   F2=Split   F3=Exit   F9=SwapNext F10=Messages F12=Cancel
```

Figure 100. Starting a system function

Select function type 1 (channel initiator), and press enter.

Stopping a channel initiator

You can stop a channel initiator using MQSC commands or using the operations and control panels.

To stop a channel initiator using the MQSC commands, use STOP CHINIT.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	7 (Stop)
Object type	SYSTEM
Name	Blank

The Stop a System Function panel is displayed. The text following the panel explains how you to use this panel:

```
Stop a System Function

Select function type, complete fields, then press Enter to stop system
function.

Function type . . . . . . . . _   1. Channel initiator
2. Channel listener
Action queue manager . . . : MQ25

Channel initiator
Restart shared channels  Y Y=Yes, N=No

Channel listener
Inbound disposition . . . Q G=Group, Q=Qmgr
Transport type . . . . . . _ L=LU6.2, T=TCP/IP

Port number (TCP/IP) . . . _____
IP address (TCP/IP) . . . ________________________________________________



Command ===> ______________________________________________________________
F1=Help   F2=Split   F3=Exit   F9=SwapNext F10=Messages F12=Cancel
```

Figure 101. Stopping a function control

Select function type 1 (channel initiator) and press enter.

The channel initiator waits for all running channels to stop in quiesce mode before it stops.

Note: If some of the channels are receiver or requester channels that are running but not active, a stop request issued to either the receiver or sender channel initiator causes it to stop immediately.

However, if messages are flowing, the channel initiator waits for the current batch of messages to complete before it stops.

Starting a channel listener

You can start a channel listener using MQSC commands or using the operations and control panels.

To start a channel listener using the MQSC commands, use START LISTENER.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	6 (Start)

Field	Value
Object type	SYSTEM
Name	Blank

The Start a System Function panel is displayed (see Figure 100 on page 597).

Select function type 2 (channel listener). Select Inbound disposition. Select Transport type. If the Transport type is L, select LU name. If the Transport type is T, select Port number and (optionally) IP address. Press enter.

Note: For the TCP/IP listener, you can start multiple combinations of Port and IP address.

Stopping a channel listener

You can stop a channel listener using MQSC commands or using the operations and control panels.

To stop a channel listener using the MQSC commands, use STOP LISTENER.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	7 (Stop)
Object type	SYSTEM
Name	Blank

The Stop a System Function panel is displayed (see Figure 101 on page 598).

Select function type 2 (channel listener). Select Inbound disposition. Select Transport type. If the transport type is 'T', select Port number and (optionally) IP address. Press enter.

Note: For a TCP/IP listener, you can stop specific combinations of Port and IP address, or you can stop all combinations.

Starting a channel

You can start a channel using MQSC commands or using the operations and control panels.

To start a channel using the MQSC commands, use START CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	6 (Start)
Object type	channel type (for example SENDER) or CHANNEL
Name	CHANNEL.TO.USE
Disposition	The disposition of the object.

The Start a Channel panel is displayed. The text following the panel explains how to use the panel:

```
Start a Channel

Select disposition, then press Enter to start channel.


Channel name . . . . . . . : CHANNEL.TO.USE
Channel type . . . . . . . : SENDER
Description . . . . . . . . : Description of CHANNEL.TO.USE


Disposition . . . . . . . . . P   P=Private on MQ25
S=Shared on MQ25
A=Shared on any queue manager








Command ===> _________________________________________________________________
F1=Help   F2=Split   F3=Exit   F9=SwapNext F10=Messages F12=Cancel
```

Figure 102. Starting a channel

Select the disposition of the channel instance and on which queue manager it is to be started.

Press enter to start the channel.

Starting a shared channel

To start a shared channel, and keep it on a nominated channel initiator, use disposition = S (on the START CHANNEL command, specify CHLDISP(FIXSHARED)).

There can be only one instance of the shared channel running at a time. Attempts to start a second instance of the channel fail.

When you start a channel in this way, the following rules apply to that channel:

- You can stop the channel from any queue manager in the queue-sharing group. You can stop it even if the channel initiator on which it was started is not running at the time you issue the stop-channel request. When the channel has stopped, you can restart it by specifying disposition = S (CHLDISP(FIXSHARED)) on the same, or another, channel initiator. You can also start it by specifying disposition = A (CHLDISP(SHARED)).
- If the channel is in the starting or retry state, you can restart it by specifying disposition = S (CHLDISP(FIXSHARED)) on the same or a different channel initiator. You can also start it by specifying disposition = A (CHLDISP(SHARED)).
- The channel is eligible to be trigger started when it goes into the inactive state. Shared channels that are trigger started always have a shared disposition (CHLDISP(SHARED)).
- The channel is eligible to be started with CHLDISP(FIXSHARED), on any channel initiator, when it goes into the inactive state. You can also start it by specifying disposition = A (CHLDISP(SHARED)).
- The channel is not recovered by any other active channel initiator in the queue-sharing group when the channel initiator on which it was started is stopped with SHARED(RESTART), or when the channel initiator terminates abnormally. The channel is recovered only when the channel initiator on which it was started is next restarted. This stops failed channel-recovery attempts being passed to other channel initiators in the queue-sharing group, which would add to their workload.

Testing a channel

You can test a channel using MQSC commands or using the operations and control panels.

To test a channel using the MQSC commands, use PING CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	5 (Perform)
Object type	SENDER, SERVER, or CHANNEL
Name	CHANNEL.TO.USE
Disposition	The disposition of the channel object.

The Perform a Channel Function panel is displayed. The text following the panel explains how to use the panel:

```
Perform a Channel Function

Select function type, complete fields, then press Enter.


Function type . . . . . . . . _   1. Reset  3. Resolve with commit
2. Ping  4. Resolve with backout

Channel name . . . . . . . : CHANNEL.TO.USE
Channel type . . . . . . . : SENDER
Description . . . . . . . . : Description of CHANNEL.TO.USE


Disposition . . . . . . . . . P   P=Private on MQ25
S=Shared on MQ25
A=Shared on any queue manager

Sequence number for reset . . 1     1 - 999999999
Data length for ping . . . . 16   16 - 32768



Command ===> _________________________________________________________________
F1=Help   F2=Split   F3=Exit   F9=SwapNext F10=Messages F12=Cancel
```

Figure 103. Testing a channel

Select function type 2 (ping).

Select the disposition of the channel for which the test is to be done and on which queue manager it is to be tested.

The data length is initially set to 16. Change it if you want and press enter.

Resetting message sequence numbers for a channel

You can reset message sequence numbers for a channel using MQSC commands or using the operations and control panels.

To reset channel sequence numbers using the MQSC commands, use RESET CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	5 (Perform)
Object type	channel type (for example SENDER) or CHANNEL
Name	CHANNEL.TO.USE
Disposition	The disposition of the channel object.

The Perform a Channel Function panel is displayed (see Figure 103 on page 601).

Select Function type 1 (reset).

Select the disposition of the channel for which the reset is to be done and on which queue manager it is to be done.

The **sequence number** field is initially set to one. Change this value if you want, and press enter.

Resolving in-doubt messages on a channel

You can resolve in-doubt messages on a channel using MQSC commands or using the operations and control panels.

To resolve in-doubt messages on a channel using the MQSC commands, use RESOLVE CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	5 (Perform)
Object type	SENDER, SERVER, or CHANNEL
Name	CHANNEL.TO.USE
Disposition	The disposition of the object.

The Perform a Channel Function panel is displayed (see Figure 103 on page 601).

Select Function type 3 or 4 (resolve with commit or backout). (See “In-doubt channels” on page 163 for more information.)

Select the disposition of the channel for which resolution is to be done and which queue manager it is to be done on. Press enter.

Stopping a channel

You can stop a channel using MQSC commands or using the operations and control panels.

To stop a channel using the MQSC commands, use STOP CHANNEL.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	Value
Action	7 (Stop)
Object type	channel type (for example SENDER) or CHANNEL
Name	CHANNEL.TO.USE
Disposition	The disposition of the object.

The Stop a Channel panel is displayed. The text following the panel explains how to use the panel:

```
Stop a Channel

Complete fields, then press Enter to stop channel.


Channel name . . . . . . . : CHANNEL.TO.USE
Channel type . . . . . . . : SENDER
Description . . . . . . . . : Description of CHANNEL.TO.USE


Disposition . . . . . . . . . P   P=Private on MQ25
A=Shared on any queue manager

Stop mode . . . . . . . . . . 1   1. Quiesce  2. Force
Stop status . . . . . . . . . 1   1. Stopped  2. Inactive

Queue manager . . . . . . . . ________________________________________________
Connection name . . . . . . . ________________________________________________




Command ===> ________________________________________________________________
F1=Help   F2=Split   F3=Exit   F9=SwapNext F10=Messages F12=Cancel
```

Figure 104. Stopping a channel

Select the disposition of the channel for which the stop is to be done and on which queue manager it is to be stopped.

Choose the stop mode that you require:

Quiesce
: The channel stops when the current message is completed and the batch is then ended, even if the batch size value has not been reached and there are messages already waiting on the transmission queue. No new batches are started. This mode is the default.

Force
: The channel stops immediately. If a batch of messages is in progress, an 'in-doubt' situation can result.

Choose the queue manager and connection name for the channel you want to stop.

Choose the status that you require:

Stopped
: The channel is not restarted automatically, and must be restarted manually. This mode is the default if no queue manager or connection name is specified. If a name is specified, it is not allowed.

Inactive
: The channel is restarted automatically when required. This mode is the default if a queue manager or connection name is specified.

Press enter to stop the channel.

See “Stopping and quiescing channels” on page 161 for more information. For information about restarting stopped channels, see “Restarting stopped channels” on page 162.

Note: If a shared channel is in a retry state and the channel initiator on which it was started is not running, a STOP request for the channel is issued on the queue manager where the command was entered.

Displaying channel status

You can display channel status by using MQSC commands, or by using the operations and control panels.

To display the status of a channel or a set of channels using the MQSC commands, use DISPLAY CHSTATUS.

Note: Displaying channel status information can take some time if you have lots of channels.

Using the operations and control panels on the List Channel panel (see Figure 97 on page 594), a summary of the channel status is shown for each channel as follows:

INACTIVE	No connections are active
status	One connection is active
nnn status	More than one connection is current and all current connections have the same status
nnn CURRENT	More than one connection is current and the current connections do not all have the same status
Blank	IBM MQ is unable to determine how many connections are active (for example, because the channel initiator is not running)

Note: For channel objects with the disposition GROUP, no status is displayed.

where *nnn* is the number of active connections, and *status* is one of the following:

INIT	INITIALIZING
BIND	BINDING
START	STARTING
RUN	RUNNING
STOP	STOPPING or STOPPED
RETRY	RETRYING
REQST	REQUESTING

To display more information about the channel status, press the Status key (F11) on the List Channel or the Display, or Alter channel panels to display the List Channels - Current Status panel (see Figure 105 on page 604).

```
List Channels - Current Status - MQ25      Row 1 of 16

Type action codes, then press Enter. Press F11 to display saved status.
1=Display current status

Channel name     Connection name                State
Start time      Messages Last message time  Type   Disposition
<> *                          CHANNEL   ALL   MQ25

_ RMA0.CIRCUIT.ACL.F  RMA1                      STOP
_  2005-03-21 10.22.36 557735  2005-03-24 09.51.11 SENDER  PRIVATE MQ25
_ RMA0.CIRCUIT.ACL.N  RMA1
_  2005-03-21 10.23.09 378675  2005-03-24 09.51.10 SENDER  PRIVATE MQ25
_ RMA0.CIRCUIT.CL.F  RMA2
_  2005-03-24 01.12.51 45544   2005-03-24 09.51.08 SENDER  PRIVATE MQ25
_ RMA0.CIRCUIT.CL.N  RMA2
_  2005-03-24 01.13.55 45560   2005-03-24 09.51.11 SENDER  PRIVATE MQ25
_ RMA1.CIRCUIT.CL.F  RMA1
_  2005-03-21 10.24.12 360757  2005-03-24 09.51.11 RECEIVER PRIVATE MQ25
_ RMA1.CIRCUIT.CL.N  RMA1
_  2005-03-21 10.23.40 302870  2005-03-24 09.51.09 RECEIVER PRIVATE MQ25
******** End of list ********
Command ===> ________________________________________________________________
F1=Help   F2=Split   F3=Exit   F4=Filter  F5=Refresh  F7=Bkwd
F8=Fwd    F9=SwapNext F10=Messages F11=Saved  F12=Cancel
```

Figure 105. Listing channel connections

The values for status are as follows:

INIT	INITIALIZING
BIND	BINDING
START	STARTING
RUN	RUNNING
STOP	STOPPING or STOPPED
RETRY	RETRYING
REQST	REQUESTING
DOUBT	STOPPED and INDOUBT(YES)

See “Channel states” on page 154 for more information.

You can press F11 to see a similar list of channel connections with saved status; press F11 to get back to the current list. The saved status does not apply until at least one batch of messages has been transmitted on the channel.

Use action code 1 or a slash (/) to select a connection and press enter. The Display Channel Connection Current Status panels are displayed.

Displaying cluster channels

You can display cluster channels using MQSC commands or using the operations and control panels.

To display all the cluster channels that have been defined (explicitly or using auto-definition), use the MQSC command, DISPLAY CLUSQMGR.

Using the operations and control panels, starting from the initial panel, complete these fields and press enter:

Field	**Value**
Action	1 (List or Display)
Object type	CLUSCHL
Name	*

You are presented with a panel like figure Figure 106 on page 606, in which the information for each cluster channel occupies three lines, and includes its channel, cluster, and queue manager names. For cluster-sender channels, the overall state is shown.

```
List Cluster-queue-manager Channels - MQ25    Row 1 of 9

Type action codes, then press Enter. Press F11 to display connection status.
1=Display  5=Perform  6=Start  7=Stop

Channel name     Connection name                 State
Type     Cluster name                    Suspended
Cluster queue manager name             Disposition
<>   *                         -    MQ25
_ TO.MQ90.T      HURSLEY.MACH90.COM(1590)
_  CLUSRCVR   VJH01T                      N
_   MQ90                       -    MQ25
_ TO.MQ95.T      HURSLEY.MACH95.COM(1595)             RUN
_  CLUSSDRA   VJH01T                      N
_   MQ95                       -    MQ25
_ TO.MQ96.T      HURSLEY.MACH96.COM(1596)             RUN
_  CLUSSDRB   VJH01T                      N
_   MQ96                       -    MQ25
******** End of list ********


Command ===> ______________________________________________________________
F1=Help   F2=Split   F3=Exit   F4=Filter  F5=Refresh  F7=Bkwd
F8=Fwd    F9=SwapNext F10=Messages F11=Status  F12=Cancel
```

Figure 106. Listing cluster channels

To display full information about one or more channels, type Action code 1 against their names and press enter. Use Action codes 5, 6, or 7 to perform functions (such as ping, resolve, and reset), and start or stop a cluster channel.

To display more information about the channel status, press the Status key (F11).

z/OS *Preparing IBM MQ for z/OS to use the zEnterprise Data Compression Express facility*

The zEnterprise® Data Compression (zEDC) Express facility is available for certain models of IBM Z® machines, starting from IBM zEC12 GA2, using a minimum z/OS level of z/OS 2.1.

See zEnterprise Data Compression (zEDC) for further information.

Prerequisites

For IBM z15 and later, the zEnterprise Data Compression (zEDC) Express facility was moved from an optional feature in the PCIe I/O drawer of the hardware system to be on-chip as the Integrated Accelerator for zEDC. With this change, the configuration prerequisites are updated and are dependent on your hardware system.

IBM z15 or later

Apply one of the following PTFs, according to your level of z/OS:

- z/OS 2.5: UJ00639
- z/OS 2.4: UJ00636
- z/OS 2.3: UJ00635
- z/OS 2.2: UJ00638

There are no hardware requirements for z15 or later systems. The Integrated Accelerator for zEDC solution in these systems provides built-in data acceleration, so a separate adapter is no longer required.

IBM zEC12 GA2 to IBM z14

Your system must also have the following requirements:

- A zEDC Express® adapter, installed in the PCIe I/O drawers of the hardware system.
- The zEDC software capability (an optional, paid-for feature) must be enabled in an IFAPRDxx parmlib member.

Procedure

IBM zEC12 GA2 to IBM z14

Ensure that the channel initiator user ID has READ authority to the FPZ.ACCELERATOR.COMPRESSION profile in the RACF FACILITY CLASS, or the equivalent in the external security manager (ESM) that your enterprise uses.

Attention: Not required for IBM z15 or later.

IBM zEnterprise zEC12 GA2 or later

Configure the channel with COMPMSG(ZLIBFAST) at both the sending and receiving ends. Once configured, zlib compression is used to compress and decompress messages flowing across the channel.

Compression is performed in the zEDC when the size of the data to be compressed is above the minimum threshold. The threshold is dependent upon the IBM z hardware being used

- IBM zEC12 GA2 to IBM z14 has a minimum threshold of 4KB
- IBM z15 or later has a minimum threshold of 1KB

For messages below the threshold size, compression or inflation is performed in the software.

Setting up communication for z/OS

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. To succeed, it is necessary for the connection to be defined and available. This section explains how to define a connection.

DQM is a remote queuing facility for IBM MQ. It provides channel control programs for the queue manager that form the interface to communication links. These links are controllable by the system operator. The channel definitions held by distributed queuing management use these connections.

Choose from one of the two forms of communication protocol that can be used for z/OS:

- “Defining a TCP connection on z/OS” on page 608
- “Defining an LU6.2 connection for z/OS using APPC/MVS” on page 610

Each channel definition must specify only one protocol as the transmission protocol (Transport Type) attribute. A queue manager can use more than one protocol to communicate.

You might also find it helpful to refer to Example configuration - IBM MQ for z/OS . If you are using queue sharing groups, see“Setting up communication for IBM MQ for z/OS using queue-sharing groups” on page 615.

Related concepts

“Using the panels and the commands” on page 593
You can use the MQSC commands, the PCF commands, or the operations and control panels to manage DQM.

“Configuring distributed queuing” on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

“Setting up communications with other queue managers” on page 587
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

“Customizing IBM MQ for z/OS” on page 526

Use this topic as a step by step guide for customizing your IBM MQ system.

“Monitoring and controlling channels on z/OS” on page 591
Use the DQM commands and panels to create, monitor, and control the channels to remote queue managers.

“Preparing IBM MQ for z/OS for DQM with queue-sharing groups” on page 611
Use the instructions in this section to configure distributed queuing with queue-sharing groups on IBM MQ for z/OS.

“Setting up communication for IBM MQ for z/OS using queue-sharing groups” on page 615
When a distributed-queuing management channel is started, it attempts to use the connection specified in the channel definition. For this attempt to succeed, it is necessary for the connection to be defined and available.

Defining a TCP connection on z/OS

To define a TCP connection, there are a number of settings to configure.

The TCP address space name must be specified in the TCP system parameters data set, *tcpip*.TCPIP.DATA. In the data set, a "TCPIPJOBNAME *TCPIP_proc*" statement must be included.

If you are using a firewall, you need to configure `allow` connections from the channel initiator to the addresses in the channels, and from the remote connections into the queue manager.

Typically the definition for a firewall configures the sending IP address and port to the destination IP address and port:

- A z/OS image can have more than one host name, and you might need to configure the firewall with multiple host addresses as the source address.

  You can use the NETSTAT HOME command to display these names and addresses.
- A channel initiator can have multiple listeners on different ports, so you need to configure these ports.
- If you are using a shared port for a queue sharing group you must configure the shared port as well.

The channel initiator address space must have authority to read the data set. The following techniques can be used to access your TCPIP.DATA data set, depending on which TCP/IP product and interface you are using:

- Environment variable, RESOLVER_CONFIG
- HFS file, /etc/resolv.conf
- //SYSTCPD DD statement
- //SYSTCPDD DD statement
- *jobname/userid*.TCPIP.DATA
- SYS1.TCPPARMS(TCPDATA)
- *zapname*.TCPIP.DATA

You must also be careful to specify the high-level qualifier for TCP/IP correctly.

You need a suitably configured Domain Name System (DNS) server, capable of both Name to IP address translation and IP address to Name translation.

Note: Some changes to the resolver configuration require a recycle of applications using it, for example, IBM MQ.

For more information, see the following:

- Base TCP/IP system
- z/OS UNIX System Services.

Each TCP channel when started uses TCP resources; you might need to adjust the following parameters in your PROFILE.TCPIP configuration data set:

ACBPOOLSIZE
Add one per started TCP channel, plus one

CCBPOOLSIZE
Add one per started TCP channel, plus one per DQM dispatcher, plus one

DATABUFFERPOOLSIZE
Add two per started TCP channel, plus one

MAXFILEPROC
Controls how many channels each dispatcher in the channel initiator can handle.

This parameter is specified in the BPXPRMxx member of SYSI.PARMLIB. Ensure that you specify a value large enough for your needs.

By default, the channel initiator is only capable of binding to IP addresses associated with the stack named in the TCPNAME queue manager attribute. To allow the channel initiator to communicate using additional TCP/IP stacks on the system, change the TCPSTACK queue manager attribute to MULTIPLE.

Related concepts

"Sending end" on page 609
At the sending end of the TCP/IP connection, there are a number of settings to configure.

"Receiving on TCP" on page 609
At the receiving end of the TCP/IP connection, there are a number of settings to configure.

"Using the TCP listener backlog option" on page 610
When receiving on TCP/IP, a maximum number of outstanding connection requests is set. These outstanding requests can be considered a *backlog* of requests waiting on the TCP/IP port for the listener to accept the request.

Sending end
At the sending end of the TCP/IP connection, there are a number of settings to configure.

The connection name (CONNAME) field in the channel definition must be set to either the host name (for example MVSHUR1) or the TCP network address of the target. The TCP network address can be in IPv4 dotted decimal form (for example 127.0.0.1) or IPv6 hexadecimal form (for example 2001:DB8:0:0:0:0:0:0). If the connection name is a host name, a TCP name server is required to convert the host name into a TCP host address. (This requirement is a function of TCP, not IBM MQ.)

On the initiating end of a connection (sender, requester, and server channel types) it is possible to provide an optional port number for the connection, for example:

Connection name
192.0.2.0(1555)

In this case the initiating end attempts to connect to a receiving program listening on port 1555.

Note: The default port number of 1414 is used if an optional port number is not specified.

The channel initiator can use any TCP/IP stack which is active and available. By default, the channel initiator binds its outbound channels to the default IP address for the TCP/IP stack named in the TCPNAME queue manager attribute. To connect through a different stack, you need to specify either the host name or IP address of the stack in the LOCLADDR attribute of the channel.

Receiving on TCP
At the receiving end of the TCP/IP connection, there are a number of settings to configure.

Receiving channel programs are started in response to a startup request from the sending channel. To do so, a listener program has to be started to detect incoming network requests and start the associated channel. You start this listener program with the START LISTENER command, or using the operations and control panels.

By default:

- The TCP Listener program uses port 1414 and listens on all addresses available to your TCP stack.

- TCP/IP listeners can bind only to addresses associated with the TCP/IP stack named in the TCPNAME queue manager attribute.

To start listeners for other addresses, or all available TCP stacks, set your TCPSTACK queue manager attribute to 'MULTIPLE'.

You can start your TCP listener program to listen only on a specific address or host name by specifying IPADDR in the START LISTENER command. For more information, see Listeners.

Using the TCP listener backlog option
When receiving on TCP/IP, a maximum number of outstanding connection requests is set. These outstanding requests can be considered a *backlog* of requests waiting on the TCP/IP port for the listener to accept the request.

The default listener backlog value on z/OS is 255. If the backlog reaches this values, the TCP/IP connection is rejected and the channel is not able to start.

For MCA channels, this results in the channel going into a RETRY state and retrying the connection at a later time.

For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and can retry the connection at a later time.

Defining an LU6.2 connection for z/OS using APPC/MVS

To define an LU6.2 connection there are a number of settings to configure.

APPC/MVS setup

Each instance of the channel initiator must have the name of the LU that it is to use defined to APPC/MVS, in the APPCPMxx member of SYS1.PARMLIB, as in the following example:

```
LUADD ACBNAME( luname ) NOSCHED TPDATA(CSQ.APPCTP)
```

luname is the name of the logical unit to be used. NOSCHED is required; TPDATA is not used. No additions are necessary to the ASCHPMxx member, or to the APPC/MVS TP profile data set.

The side information data set must be extended to define the connections used by DQM. See the supplied sample CSQ4SIDE for details of how to do this using the APPC utility program ATBSDFMU. For details of the TPNAME values to use, see the following table for information:

Table 48. Settings on the local z/OS system for a remote queue manager platform

Remote platform	TPNAME
z/OS or MVS	The same as TPNAME in the corresponding side information about the remote queue manager.
IBM i	The same as the compare value in the routing entry on the IBM i system.
HP Integrity NonStop Server	The same as the TPNAME specified in the receiver-channel definition.
UNIX and Linux systems	The same as TPNAME in the corresponding side information about the remote queue manager.
Windows	As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.

If you have more than one queue manager on the same machine, ensure that the TPnames in the channel definitions are unique.

See the *Multiplatform APPC Configuration Guide* also for information about the VTAM definitions that might be required.

In an environment where the queue manager is communicating using APPC with a queue manager on the same or another z/OS system, ensure that either the VTAM definition for the communicating LU specifies SECACPT(ALREADYV), or that there is a RACF APPCLU profile for the connection between LUs, which specifies CONVSEC(ALREADYV).

The z/OS command VARY ACTIVE must be issued against both base and listener LUs before attempting to start either inbound or outbound communications.

Attention: In addition to the APPC setup, you must issue the following command:

```
ALTER QMGR LUNAME(luname)
```

and restart the channel initiator.

See LUNAME for further information.

Related concepts

"Connecting to LU 6.2" on page 611
To connect to LU 6.2, there are a number of settings to configure.

"Receiving on LU 6.2" on page 611
To receive on LU 6.2, there are a number of settings to configure.

Connecting to LU 6.2
To connect to LU 6.2, there are a number of settings to configure.

The connection name (CONNAME) field in the channel definition must be set to the symbolic destination name, as specified in the side information data set for APPC/MVS.

The LU name to use (defined to APPC/MVS as described previously) must also be specified in the channel initiator parameters. It must be set to the same LU that is used for receiving by the listener.

The channel initiator uses the "SECURITY(SAME)" APPC/MVS option, so it is the user ID of the channel initiator address space that is used for outbound transmissions, and is presented to the receiver.

Receiving on LU 6.2
To receive on LU 6.2, there are a number of settings to configure.

Receiving MCAs are started in response to a startup request from the sending channel. To do so, a listener program has to be started to detect incoming network requests and start the associated channel. The listener program is an APPC/MVS server. You start it with the START LISTENER command, or using the operations and control panels. You must specify the LU name to use with a symbolic destination name defined in the side information data set. The local LU so identified must be the same as the one used for outbound transmissions, as set in the channel initiator parameters.

Preparing IBM MQ for z/OS for DQM with queue-sharing groups

Use the instructions in this section to configure distributed queuing with queue-sharing groups on IBM MQ for z/OS.

For an example configuration using queue-sharing groups, see Example configuration - IBM MQ for z/OS using queue-sharing groups. For a message channel planning example using queue-sharing groups, see Message channel planning example for z/OS using queue-sharing groups.

You need to create and configure the following components to enable distributed queuing with queue-sharing groups:

- LU 6.2 and TCP/IP listeners
- Transmission queues and triggering
- Message channel agents
- Synchronization queue

After you have created the components you need to set up the communication, see "Setting up communication for IBM MQ for z/OS using queue-sharing groups" on page 615.

For information about how to monitor and control channels when using queue-sharing groups, see "Monitoring and controlling channels on z/OS" on page 591.

See the following sections for queue-sharing group concepts and benefits.

Class of service

A shared queue is a type of local queue that offers a different class of service. Messages on a shared queue are stored in a coupling facility (CF), which allows them to be accessed by all queue managers in the queue-sharing group. A message on a shared queue must be a message of length no more than 100 MB.

Generic interface

A queue-sharing group has a generic interface that allows the network to view the group as a single entity. This view is achieved by having a single generic address that can be used to connect to any queue manager within the group.

Each queue manager in the queue-sharing group listens for inbound session requests on an address that is logically related to the generic address. For more information see "LU 6.2 and TCP/IP listeners for queue-sharing groups" on page 613.

Load-balanced channel start

A shared transmission queue can be serviced by an outbound channel running on any channel initiator in the queue-sharing group. Load-balanced channel start determines where a start channel command is targeted. An appropriate channel initiator is chosen that has access to the necessary communications subsystem. For example, a channel defined with TRPTYPE(LU6.2) cannot be started on a channel initiator that only has access to a TCP/IP subsystem.

The choice of channel initiator is dependent on the channel load and the headroom of the channel initiator. The channel load is the number of active channels as a percentage of the maximum number of active channels allowed as defined in the channel initiator parameters. The headroom is the difference between the number of active channels and the maximum number allowed.

Inbound shared channels can be load-balanced across the queue-sharing group by use of a generic address, as described in "LU 6.2 and TCP/IP listeners for queue-sharing groups" on page 613.

Shared channel recovery

The following table shows the types of shared-channel failure and how each type is handled.

Type of failure:	**What happens:**
Channel initiator communications subsystem failure	The channels dependent on the communications subsystem enter channel retry, and are restarted on an appropriate queue-sharing group channel initiator by a load-balanced start command.
Channel initiator failure	The channel initiator fails, but the associated queue manager remains active. The queue manager monitors the failure and initiates recovery processing.
Queue manager failure	The queue manager fails (failing the associated channel initiator). Other queue managers in the queue-sharing group monitor the event and initiate peer recovery.
Shared status failure	Channel state information is stored in Db2, so a loss of connectivity to Db2 becomes a failure when a channel state change occurs. Running channels can carry on running without access to these resources. On a failed access to Db2, the channel enters retry.

Shared channel recovery processing on behalf of a failed system requires connectivity to Db2 to be available on the system managing the recovery to retrieve the shared channel status.

Client channels

Client connection channels can benefit from the high availability of messages in queue-sharing groups that are connected to the generic interface instead of being connected to a specific queue manager. For more information, see Client connection channels.

Related concepts

Shared queues and queue-sharing groups

“Customizing IBM MQ for z/OS” on page 526
Use this topic as a step by step guide for customizing your IBM MQ system.

“Configuring distributed queuing” on page 122
This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

“Setting up communications with other queue managers” on page 587
This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

“Clusters and queue-sharing groups” on page 615
You can make your shared queue available to a cluster in a single definition. To do so you specify the name of the cluster when you define the shared queue.

“Channels and serialization” on page 615
During shared queue peer recovery, message channel agents that process messages on shared queues serialize their access to the queues.

Intra-group queuing

LU 6.2 and TCP/IP listeners for queue-sharing groups

The group LU 6.2 and TCP/IP listeners listen on an address that is logically connected to the generic address.

For the LU 6.2 listener, the specified LUGROUP is mapped to the VTAM generic resource associated with the queue-sharing group. For an example of setting up this technology, see “Defining an LU6.2 connection for z/OS using APPC/MVS” on page 610.

For the TCP/IP listener, the specified port can be connected to the generic address in one of the following ways:

- For a front-end router such as the IBM Network Dispatcher, inbound connect requests are forwarded from the router to the members of the queue-sharing group.
- For TCP/IP Sysplex Distributor, each listener that is running and is listening on a particular address that is set up as a Distributed DVIPA is allocated a proportion of the incoming requests. For an example of setting up this technology, see Using Sysplex Distributor

z/OS *Transmission queues and triggering for queue-sharing groups*

A shared transmission queue is used to store messages before they are moved from the queue-sharing group to the destination.

It is a shared queue and it is accessible to all queue managers in the queue-sharing group.

Triggering

A triggered shared queue can generate more than one trigger message for a satisfied trigger condition. There is one trigger message generated for each local initiation queue defined on a queue manager in the queue-sharing group associated with the triggered shared queue.

For distributed queuing, each channel initiator receives a trigger message for a satisfied shared transmission queue trigger condition. However, only one channel initiator actually processes the triggered start, and the others fail safely. The triggered channel is then started with a load balanced start (see “Preparing IBM MQ for z/OS for DQM with queue-sharing groups” on page 611) that is triggered to start channel `QSG.TO.QM2`. To create a shared transmission queue, use the IBM MQ commands (MQSC) as shown in the following example:

```
DEFINE QLOCAL(QM2) DESCR('Transmission queue to QM2') +
USAGE(XMITQ) QSGDISP(SHARED) +
CFSTRUCT(APPLICATION1) INITQ(SYSTEM.CHANNEL.INITQ) +
TRIGGER TRIGDATA(QSG.TO.QM2)
```

Message channel agents for queue-sharing groups

A channel can only be started on a channel initiator if it has access to a channel definition for a channel with that name.

A message channel agent is an IBM MQ program that controls the sending and receiving of messages. Message channel agents move messages from one queue manager to another; there is one message channel agent at each end of a channel.

A channel definition can be defined to be private to a queue manager or stored on the shared repository and available anywhere (a group definition). This means that a group defined channel is available on any channel initiator in the queue-sharing group.

Note: The private copy of the group definition can be changed or deleted.

To create group channel definitions, use the IBM MQ commands (MQSC) as shown in the following examples:

```
DEFINE CHL(QSG.TO.QM2) CHLTYPE(SDR) +
TRPTYPE(TCP) CONNAME(QM2.MACH.IBM.COM) +
XMITQ(QM2) QSGDISP(GROUP)
```

```
DEFINE CHL(QM2.TO.QSG) CHLTYPE(RCVR) TRPTYPE(TCP) +
QSGDISP(GROUP)
```

There are two perspectives from which to look at the message channel agents used for distributed queuing with queue-sharing groups:

Inbound

An inbound channel is a shared channel if it is connected to the queue manager through the group listener. It is connected either through the generic interface to the queue-sharing group, then directed to a queue manager within the group, or targeted at the group port of a specific queue manager or the luname used by the group listener.

Outbound

An outbound channel is a shared channel if it moves messages from a shared transmission queue. In the example commands, sender channel `QSG.TO.QM2` is a shared channel because its transmission queue, QM2 is defined with QSGDISP(SHARED).

Synchronization queue for queue-sharing groups

Shared channels have their own shared synchronization queue called SYSTEM.QSG.CHANNEL.SYNCQ.

This synchronization queue is accessible to any member of the queue-sharing group. (Private channels continue to use the private synchronization queue. See “Defining IBM MQ objects” on page 589). This means that the channel can be restarted on a different queue manager and channel initiator instance within the queue-sharing group in the event of failure of the communications subsystem, channel initiator, or queue manager. For further information, see “Preparing IBM MQ for z/OS for DQM with queue-sharing groups” on page 611.

DQM with queue-sharing groups requires that a shared queue is available with the name SYSTEM.QSG.CHANNEL.SYNCQ. This queue must be available so that a group listener can successfully start.

If a group listener fails because the queue was not available, the queue can be defined and the listener can be restarted without recycling the channel initiator. The non-shared channels are not affected.

Make sure that you define this queue using INDXTYPE(MSGID). This definition improves the speed at which messages on the queue can be accessed.

Clusters and queue-sharing groups

You can make your shared queue available to a cluster in a single definition. To do so 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 all members of the queue-sharing group to put messages to the same shared queue.

For more information, see “Configuring a queue manager cluster” on page 215.

Channels and serialization

During shared queue peer recovery, message channel agents that process messages on shared queues serialize their access to the queues.

If a queue manager in a queue-sharing group fails while a message channel agent is dealing with uncommitted messages on one or more shared queues, the channel and the associated channel initiator will end, and shared queue peer recovery will take place for the queue manager.

Because shared queue peer recovery is an asynchronous activity, peer channel recovery might try to simultaneously restart the channel in another part of the queue sharing group before shared queue peer recovery is complete. If this event happens, committed messages might be processed ahead of the messages still being recovered. To ensure that messages are not processed out of sequence in this way, message channel agents that process messages on shared queues serialize their access to these queues.

An attempt to start a channel for which shared queue peer recovery is still in progress might result in a failure. An error message indicating that recovery is in progress is issued, and the channel is put into retry state. Once queue manager peer recovery is complete, the channel can restart at the time of the next retry.

An attempt to RESOLVE, PING, or DELETE a channel can fail for the same reason.

Setting up communication for IBM MQ for z/OS using queue-sharing groups

When a distributed-queuing management channel is started, it attempts to use the connection specified in the channel definition. For this attempt to succeed, it is necessary for the connection to be defined and available.

Choose from one of the two forms of communication protocol that can be used:

- TCP
- LU 6.2 through APPC/MVS

You might find it useful to refer to Example configuration - IBM MQ for z/OS using queue-sharing groups.

Defining a TCP connection for queue-sharing groups

To define a TCP connection for a queue-sharing group, certain attributes on the sending and receiving end must be configured.

For information about setting up your TCP, see “Defining a TCP connection on z/OS” on page 608.

Sending end

The connection name (CONNAME) field in the channel definition to connect to your queue-sharing group must be set to the generic interface of your queue-sharing group (see Queue-sharing groups). For more details, refer to Using Sysplex Distributor.

Receiving on TCP using a queue-sharing group

Receiving shared channel programs are started in response to a startup request from the sending channel. To do so, a listener must be started to detect incoming network requests and start the associated channel. You start this listener program with the START LISTENER command, using the inbound disposition of the group, or using the operations and control panels.

All group listeners in the queue-sharing group must be listening on the same port. If you have more than one channel initiator running on a single MVS image you can define virtual IP addresses and start your TCP listener program to only listen on a specific address or host name by specifying IPADDR in the START LISTENER command. (For more information, see START LISTENER.)

Defining an LU 6.2 connection on z/OS
To define an LU 6.2 connection for a queue-sharing group, certain attributes on the sending and receiving end must be configured.

For information about setting up APPC/MVS, see Setting up communication for z/OS .

Connecting to APPC/MVS (LU 6.2)

The connection name (CONNAME) field in the channel definition to connect to your queue-sharing group must be set to the symbolic destination name, as specified in the side information data set for APPC/MVS. The partner LU specified in this symbolic destination must be the generic resource name. For more details, see Defining yourself to the network using generic resources.

Receiving on LU 6.2 using a generic interface

Receiving shared MCAs are started in response to a startup request from the sending channel. To do so, a group listener program must be started to detect incoming network requests and start the associated channel. The listener program is an APPC/MVS server. You start it with the START LISTENER command, using an inbound disposition group, or using the operations and control panels. You must specify the LU name to use a symbolic destination name defined in the side information data set. For more details, see Defining yourself to the network using generic resources.

Using IBM MQ with IMS

The IBM MQ -IMS adapter, and the IBM MQ - IMS bridge are the two components which allow IBM MQ to interact with IMS.

To configure IBM MQ and IMS to work together, you must complete the following tasks:

- “Setting up the IMS adapter” on page 617
- “Setting up the IMS bridge” on page 623

Related concepts

IBM MQ and IMS

“Configuring queue managers on z/OS” on page 522
Use these instructions to configure queue managers on IBM MQ for z/OS.

“Using IBM MQ with CICS” on page 624
To use IBM MQ with CICS, you must configure the IBM MQ CICS adapter and, optionally, the IBM MQ CICS bridge components.

“Using OTMA exits in IMS” on page 626

Use this topic if you want to use IMS Open Transaction Manager Access exits with IBM MQ for z/OS.

IMS and IMS bridge applications on IBM MQ for z/OS

Related reference

"Upgrading and applying service to Language Environment or z/OS Callable Services" on page 624
The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

Setting up the IMS adapter

To use IBM MQ within IMS requires the IBM MQ - IMS adapter (generally referred to as the IMS adapter).

This topic tells you how to make the IMS adapter available to your IMS subsystem. If you are not familiar with tailoring an IMS subsystem, see the *IMS information in IBM Documentation*.

To make the IMS adapter available to IMS applications, follow these steps:

1. Define IBM MQ to IMS as an external subsystem using the IMS external subsystem attach facility (ESAF).

   See "Defining IBM MQ to IMS" on page 618.
2. Include the IBM MQ load library thlqual.SCSQAUTH in the JOBLIB or STEPLIB concatenation in the JCL for your IMS control region and for any dependent region that connects to IBM MQ (if it is not in the LPA or link list). If your JOBLIB or STEPLIB is not authorized, also include it in the DFSESL concatenation after the library containing the IMS modules (usually IMS RESLIB).

   Also include thlqual.SCSQANLx (where x is the language letter).

   If DFSESL is present, then SCSQAUTH and SCSQANLx need to be included in the concatenation or added to LNKLIST. Adding to the STEPLIB or JOBLIB concatenation in the JCL is not sufficient.
3. Copy the IBM MQ assembler program CSQQDEFV from thlqual.SCSQASMS to a user library.
4. The supplied program, CSQQDEFV, contains one subsystem name CSQ1 identified as default with an IMS language interface token (LIT) of MQM1. You can retain this name for testing and installation verification.

   For production subsystems, you change the NAME=CSQ1 to your own subsystem name, or use CSQ1. You can add further subsystem definitions as required. See "Defining IBM MQ queue managers to the IMS adapter" on page 621 for further information on LITs.
5. Assemble and link-edit the program to produce the CSQQDEFV load module. For the assembly, include the library thlqual.SCSQMACS in your SYSLIB concatenation; use the link-edit parameter RENT. This is shown in the sample JCL in thlqual.SCSQPROC(CSQ4DEFV).
6. Include the user library containing the module CSQQDEFV that you created in the JOBLIB or STEPLIB concatenation in the JCL for any dependent region that connects to IBM MQ. Put this library before the SCSQAUTH because SCSQAUTH has a default load module. If you do not do this, you will receive a user 3041 abend from IMS.
7. If the IMS adapter detects an unexpected IBM MQ error, it issues a z/OS SNAP dump to DD name CSQSNAP and issues reason code MQRC_UNEXPECTED_ERROR to the application. If the CSQSNAP DD statement was not in the IMS dependent region JCL, no dump is taken. If this happens, you could include the CSQSNAP DD statement in the JCL and rerun the application. However, because some problems might be intermittent, it is recommended that you include the CSQSNAP DD statement to capture the reason for failure at the time it occurs.
8. If you want to use dynamic IBM MQ calls (described in Dynamically calling the IBM MQ stub), build the dynamic stub, as shown in Figure 107 on page 618.
9. If you want to use the IMS trigger monitor, define the IMS trigger monitor application CSQQTRMN, and perform PSBGEN and ACBGEN. See "Setting up the IMS trigger monitor" on page 622.
10. If you are using RACF to protect resources in the OPERCMDS class, ensure that the userid associated with your IBM MQ queue manager address space has authority to issue the MODIFY command to any IMS system to which it might connect.

```
//DYNSTUB EXEC PGM=IEWL,PARM='RENT,REUS,MAP,XREF'
//SYSPRINT DD  SYSOUT=*
//ACSQMOD  DD  DISP=SHR,DSN=thlqual.SCSQLOAD
//IMSLIB   DD  DISP=SHR,DSN=ims.reslib
//SYSLMOD  DD  DISP=SHR,DSN=private.load¹
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,1)
//SYSLIN   DD  *
  INCLUDE ACSQMOD(CSQQSTUB)
  INCLUDE IMSLIB(DFSLI000)
  ALIAS MQCONN,MQCONNX,MQDISC             MQI entry points
  ALIAS MQGET,MQPUT,MQPUT1                MQI entry points
  ALIAS MQOPEN,MQCLOSE                    MQI entry points
  ALIAS MQBACK,MQCMIT                     MQI entry points
  ALIAS CSQBBAK,CSQBCMT                   MQI entry points
  ALIAS MQINQ,MQSET                       MQI entry points
  ALIAS DFSPLI,PLITDLI                    IMS entry points
  ALIAS DFSCOBOL,CBLTDLI                  IMS entry points
  ALIAS DFSFOR,FORTDLI                    IMS entry points
  ALIAS DFSASM,ASMTDLI                    IMS entry points
  ALIAS DFSPASCL,PASTDLI                  IMS entry points
  ALIAS DFHEI01,DFHEI1                    IMS entry points
  ALIAS DFSAIBLI,AIBTDLI                  IMS entry points
  ALIAS DFSESS,DSNWLI,DSNHLI              IMS entry points
  ALIAS MQCRTMH,MQDLTMH,MQDLTMP           IMS entry points
  ALIAS MQINQMP,MQSETMP,MQMHBUF,MQBUFMH   IMS entry points
  MODE AMODE(31),RMODE(24)                Note RMODE setting
  NAME CSQQDYNS(R)
/*

¹Specify the name of a library accessible to IMS applications that
want to make dynamic calls to WebSphere MQ.
```

Figure 107. Sample JCL to link-edit the dynamic call stub

Related concepts

IBM MQ and IMS

"Setting up the IMS bridge" on page 623

The IBM MQ - IMS bridge is an optional component that enables IBM MQ to input and output to and from existing programs and transactions that are not IBM MQ-enabled.

IMS and IMS bridge applications on IBM MQ for z/OS

Defining IBM MQ to IMS

IBM MQ must be defined to the IMS control region, and to each dependent region accessing that IBM MQ queue manager. To do this, you must create a subsystem member (SSM) in the IMS.PROCLIB library, and identify the SSM to the applicable IMS regions.

Placing the subsystem member entry in IMS.PROCLIB

Each SSM entry in IMS.PROCLIB defines a connection from an IMS region to a different queue manager.

To name an SSM, concatenate the value (one to four alphanumeric characters) of the IMSID field of the IMS IMSCTRL macro with any name (one to four alphanumeric characters) defined by your site.

One SSM can be shared by all the IMS regions, or a specific member can be defined for each region. This member contains as many entries as there are connections to external subsystems. Each entry is an 80-character record.

Positional parameters

The fields in this entry are:

```
SSN,LIT,ESMT,RTT,REO,CRC
```

where:

SSN
: Specifies the IBM MQ queue manager name. It is required, and must contain one through four characters.

LIT
: Specifies the language interface token (LIT) supplied to IMS. This field is required, its value must match one in the CSQQDEFV module.

ESMT
: Specifies the external subsystem module table (ESMT). This table specifies which attachment modules must be loaded by IMS. CSQQESMT is the required value for this field.

RTT
: This option is not supported by IBM MQ.

REO
: Specifies the region error option (REO) to be used if an IMS application references a non-operational external subsystem or if resources are unavailable at create thread time. This field is optional and contains a single character, which can be:

  R
  : Passes a return code to the application, indicating that the request for IBM MQ services failed.

  Q
  : Ends the application with an abend code U3051, backs out activity to the last commit point, does a PSTOP of the transaction, and requeues the input message. This option only applies when an IMS application tries to reference a non-operational external subsystem or if the resources are unavailable at create thread time.

    IBM MQ completion and reason codes are returned to the application if the IBM MQ problem occurs while IBM MQ is processing the request; that is, after the adapter has passed the request on to IBM MQ.

  A
  : Ends the application with an abend code of U3047 and discards the input message. This option only applies when an IMS application references a non-operational external subsystem or if the resources are unavailable at create thread time.

    IBM MQ completion and reason codes are returned to the application if the IBM MQ problem occurs while IBM MQ is processing the request; that is, after the adapter has passed the request on to IBM MQ.

CRC
: This option can be specified but is not used by IBM MQ.

  Note: For full details of all positional parameters refer to How external subsystems are specified to IMS.

An example SSM entry is:

```
CSQ1,MQM1,CSQQESMT,,R,
```

where:

CSQ1	The default subsystem name as supplied with IBM MQ. You can change this to suit your installation.

MQM1 The default LIT as supplied in CSQQDEFV.

CSQQESMT The external subsystem module name. You must use this value.

R REO option.

Keyword parameters

IBM MQ parameters can be specified in keyword format; to do this you must specify SST=Db2. Other parameters are as described in Positional parameters, and shown in the following example:

```
SST=DB2,SSN=SYS3,LIT=MQM3,ESMT=CSQQESMT
```

where:

SYS3 The subsystem name

MQM3 The LIT as supplied in CSQQDEFV

CSQQESMT The external subsystem module name

Specifying the SSM EXEC parameter

Specify the SSM EXEC parameter in the startup procedure of the IMS control region. This parameter specifies the one-character to four-character subsystem member name (SSM).

If you specify the SSM for the IMS control region, any dependent region running under the control region can attach to the IBM MQ queue manager named in the IMS.PROCLIB member specified by the SSM parameter. The IMS.PROCLIB member name is the IMS ID (IMSID= *xxxx*) concatenated with the one to four characters specified in the SSM EXEC parameter. The IMS ID is the IMSID parameter of the IMSCTRL generation macro.

IMS lets you define as many external subsystem connections as are required. More than one connection can be defined for different IBM MQ queue managers. All IBM MQ connections must be within the same z/OS system. For a dependent region, you can specify a dependent region SSM or use the one specified for the control region. You can specify different region error options (REOs) in the dependent region SSM and the control region SSM. Table 49 on page 620 shows the different possibilities of SSM specifications.

Table 49. SSM specifications options

SSM for control region	SSM for dependent region	Action	Comments
No	No	None	No external subsystem can be connected.
No	Yes	None	No external subsystem can be connected.
Yes	No	Use the control region SSM	Applications scheduled in the region can access external subsystems identified in the control region SSM. Exits and control blocks for each attachment are loaded into the control region and the dependent region address spaces.
Yes	Yes (empty)	No SSM is used for the dependent region	Applications scheduled in this region can access DL/I databases only. Exits and control blocks for each attachment are loaded into the control region address space.

Table 49. SSM specifications options (continued)			
SSM for control region	**SSM for dependent region**	**Action**	**Comments**
Yes	Yes (not empty)	Check the dependent region SSM with the control region SSM	Applications scheduled in this region can access only external subsystems identified in both SSMs. Exits and control blocks for each attachment are loaded into the control region and the dependent region address spaces.

There is no specific parameter to control the maximum number of SSM specification possibilities.

Preloading the IMS adapter

The performance of the IMS adapter can be improved if it is preloaded by IMS. Preloading is controlled by the DFSMPLxx member of IMS.PROCLIB: see " IMS Administration Guide: System" for more information. The IBM MQ module names to specify are:

CSQACLST	CSQAMLST	CSQAPRH	CSQAVICM	CSQFSALM	CSQQDEFV
CSQQCONN	CSQQDISC	CSQQTERM	CSQQINIT	CSQQBACK	CSQQCMMT
CSQQESMT	CSQQPREP	CSQQTTHD	CSQQWAIT	CSQQNORM	CSQQSSOF
CSQQSSON	CSQFSTAB	CSQQRESV	CSQQSNOP	CSQQCMND	CSQQCVER
CSQQTMID	CSQQTRGI	CSQQCON2	CSQBPAPI	CSQBCRMH	CSQBAPPL

For more information on the use of IBM MQ classes for JMS, see Using IBM MQ classes for JMS in IMS.

Current releases of IMS support preloading IBM MQ modules from PDS-E format libraries in MPP, BMP, IFP, JMP and JBP regions only. Any other type of IMS region does not support preloading from PDS-E libraries. If preloading is required for any other type of region, then the IBM MQ modules that are provided must be copied to a PDS format library.

Defining IBM MQ queue managers to the IMS adapter

The names of the IBM MQ queue managers and their corresponding language interface tokens (LITs) must be defined in the queue manager definition table.

Use the supplied CSQQDEFX macro to create the CSQQDEFV load module. Figure 108 on page 621 shows the syntax of this assembler macro.

```
CSQQDEFX TYPE=ENTRY|DEFAULT,NAME=qmgr-name,LIT=token
or
CSQQDEFX TYPE=END
```

Figure 108. CSQQDEFX macro syntax

Parameters

TYPE=ENTRY|DEFAULT
Specify either TYPE=ENTRY or TYPE=DEFAULT as follows:

TYPE=ENTRY
Specifies that a table entry describing an IBM MQ queue manager available to an IMS application is to be generated. If this is the first entry, the table header is also generated, including a CSQQDEFV CSECT statement.

TYPE=DEFAULT
As for TYPE=ENTRY. The queue manager specified is the default queue manager to be used when MQCONN or MQCONNX specifies a name that is all blanks. There must be only one such entry in the table.

NAME= ***qmgr-name***
Specifies the name of the queue manager, as specified with MQCONN or MQCONNX .

LIT= ***token***
Specifies the name of the language interface token (LIT) that IMS uses to identify the queue manager.

An MQCONN or MQCONNX call associates the *name* input parameter and the *hconn* output parameter with the name label and, therefore, the LIT in the CSQQDEFV entry. Further IBM MQ calls passing the *hconn* parameter use the LIT from the CSQQDEFV entry identified in the MQCONN or MQCONNX call to direct calls to the IBM MQ queue manager defined in the IMS SSM PROCLIB member with that same LIT.

In summary, the *name* parameter on the MQCONN or MQCONNX call identifies a LIT in CSQQDEFV and the same LIT in the SSM member identifies an IBM MQ queue manager. (For information about the MQCONN call, see MQCONN - Connect queue manager. For information about the MQCONNX call, see MQCONNX - Connect queue manager (extended).)

TYPE=END
Specifies that the table is complete. If this parameter is omitted, TYPE=ENTRY is assumed.

Using the CSQQDEFX macro

Figure 109 on page 622 shows the general layout of a queue manager definition table.

```
CSQQDEFX NAME=subsystem1,LIT=token1
CSQQDEFX NAME=subsystem2,LIT=token2,TYPE=DEFAULT
CSQQDEFX NAME=subsystem3,LIT=token3
...
CSQQDEFX NAME=subsystemN,LIT=tokenN
CSQQDEFX TYPE=END
END
```

Figure 109. Layout of a queue manager definition table

Setting up the IMS trigger monitor

You can set up an IMS batch-oriented program to monitor an IBM MQ initiation queue.

Define the application to IMS using the model CSQQTAPL in the thlqual.SCSQPROC library (see Example transaction definition for CSQQTRMN).

Generate the PSB and ACB using the model CSQQTPSB in the thlqual.SCSQPROC library (see Example PSB definition for CSQQTRMN).

```
*  This is the application definition *
*  for the IMS Trigger Monitor BMP    *

APPLCTN PSB=CSQQTRMN,
PGMTYPE=BATCH,
SCHDTYP=PARALLEL
```

Figure 110. Example transaction definition for CSQQTRMN

```
PCB   TYPE=TP,       ALTPCB for transaction messages
MODIFY=YES,          To "triggered" IMS transaction
PCBNAME=CSQQTRMN
PCB   TYPE=TP,       ALTPCB for diagnostic messages
MODIFY=YES,          To LTERM specified or "MASTER"
PCBNAME=CSQQTRMG,
EXPRESS=YES
PSBGEN LANG=ASSEM,
PSBNAME=CSQQTRMN,    Runs program CSQQTRMN
CMPAT=YES
```

Figure 111. Example PSB definition for CSQQTRMN

For further information about starting and stopping the IMS trigger monitor, see Controlling the IMS trigger monitor.

Setting up the IMS bridge

The IBM MQ - IMS bridge is an optional component that enables IBM MQ to input and output to and from existing programs and transactions that are not IBM MQ-enabled.

This topic describes what you must do to customize the IBM MQ - IMS bridge.

Define the XCF and OTMA parameters for IBM MQ.

This step defines the XCF group and member names for your IBM MQ system, and other OTMA parameters. IBM MQ and IMS must belong to the same XCF group. Use the OTMACON keyword of the CSQ6SYSP macro to tailor these parameters in the system parameter load module.

See Using CSQ6SYSP for more information.

Define the XCF and OTMA parameters to IMS.

This step defines the XCF group and member names for the IMS system. IMS and IBM MQ must belong to the same XCF group.

Add the following parameters to your IMS parameter list, either in your JCL or in member DFSPBxxx in the IMS PROCLIB:

OTMA=Y

This starts OTMA automatically when IMS is started. (It is optional, if you specify OTMA=N you can also start OTMA by issuing the IMS command /START OTMA.)

GRNAME=

This parameter gives the XCF group name.

It is the same as the group name specified in the storage class definition (see the next step), and in the `Group` parameter of the OTMACON keyword of the CSQ6SYSP macro.

OTMANM=

This parameter gives the XCF member name of the IMS system.

This is the same as the member name specified in the storage class definition (see the next step).

Tell IBM MQ the XCF group and member name of the IMS system.

This is specified by the storage class of a queue. If you want to send messages across the IBM MQ - IMS bridge you must specify this when you define the storage class for the queue. In the storage class, you must define the XCF group and the member name of the target IMS system. To do this, either use the IBM MQ operations and control panels, or use the IBM MQ commands as described in Introduction to Programmable Command Formats.

Set up the security that you require.

The `/SECURE OTMA` IMS command determines the level of security to be applied to **every** IBM MQ queue manager that connects to IMS through OTMA. See Security considerations for using IBM MQ with IMS for more information.

Adding an additional IMS connection to the same queue manager

To add an IMS connection to the same queue manager you must:

- Define a second storage class STGCLASS to point at the new IMS; see DEFINE STGCLASS for more information.
- Add a new local queue to point to the second storage class.

Important:

- One local queue cannot point to two storage classes.
- One storage class cannot point to two IMS bridges.
- IBM MQ and IMS must belong to the same XCF group. Use the OTMACON keyword of the CSQ6SYSP macro to tailor these parameters in the system parameter load module.

  See Using CSQ6SYSP for more information.

Related concepts

IBM MQ and IMS

"Setting up the IMS adapter" on page 617
To use IBM MQ within IMS requires the IBM MQ - IMS adapter (generally referred to as the IMS adapter).

IMS and IMS bridge applications on IBM MQ for z/OS

Using IBM MQ with CICS

To use IBM MQ with CICS, you must configure the IBM MQ CICS adapter and, optionally, the IBM MQ CICS bridge components.

For more information about configuring the IBM MQ CICS adapter and the IBM MQ CICS bridge components, see the Configuring connections to MQ section of the CICS documentation.

Related concepts

IBM MQ and CICS

"Using IBM MQ with IMS" on page 616
The IBM MQ -IMS adapter, and the IBM MQ - IMS bridge are the two components which allow IBM MQ to interact with IMS.

Related reference

"Upgrading and applying service to Language Environment or z/OS Callable Services" on page 624
The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

Upgrading and applying service to Language Environment or z/OS Callable Services

The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

The following tables show you what you need to do to IBM MQ for z/OS if you upgrade your level of, or apply service to, the following products:

- Language Environment
- z/OS Callable Services (APPC and RRS for example)

<table>
<caption>Table 50. Service has been applied or the product has been upgraded to a new release</caption>
<tr><th>Product</th><th>Action if using CALLLIBS and SMP/E V3r2 or later

Note: You do not need to run separate jobs for Language Environment and Callable services. One job will suffice.</th><th>Action if using LINK</th></tr>
<tr><td>Language Environment</td><td>1. Set the Boundary on your SMP/E job to the Target zone.
2. On the SMPCNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See SMP/E for z/OS: Commands for further information.
3. Run the SMP/E job.</td><td>No action required provided that the SMP/E zones were set up for automatic relinking, and the CSQ8SLDQ job has been run.</td></tr>
<tr><td>Callable Services</td><td>1. Set the Boundary on your SMP/E job to the Target zone.
2. On the SMPCNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See SMP/E for z/OS: Commands for further information.
3. Run the SMP/E job.</td><td>No action required provided that the SMP/E zones were set up for automatic relinking, and the CSQ8SLDQ job has been run.</td></tr>
</table>

<table>
<caption>Table 51. One of the products has been updated to a new release in a new SMP/E environment and libraries</caption>
<tr><th>Product</th><th>Action if using CALLLIBS and SMP/E V3r2 or later

Note: You do not need to run three separate jobs for Language Environment and Callable services. One job will suffice for both products.</th><th>Action if using LINK</th></tr>
<tr><td>Language Environment</td><td>1. Change the DDDEFs for SCEELKED and SCEESPC to point to the new library.
2. Set the Boundary on your SMP/E job to the Target zone.
3. On the SMPCNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See SMP/E for z/OS: Commands for further information.
4. Run the SMP/E job.</td><td>1. Delete the XZMOD subentries for the following LMOD entries in the IBM MQ for z/OS target zone:

CMQXDCST, CMQXRCTL, CMQXSUPR, CSQCBE00, CSQCBE30, CSQCBP00, CSQCBP10, CSQCBR00, CSQUCVX, CSQUDLQH, CSQVXPCB, CSQVXSPT, CSQXDCST, CSQXRCTL, CSQXSUPR, CSQXTCMI, CSQXTCP, CSQXTNSV, CSQ7DRPS, IMQB23IC, IMQB23IM, IMQB23IR, IMQS23IC, IMQS23IM, IMQS23IR
2. Set up the appropriate ZONEINDEXs between the IBM MQ zones and the Language Environment zones.
3. Tailor CSQ8SLDQ to refer to the new zone on the FROMZONE parameter of the LINK commands. CSQ8SLDQ can be found in the SCSQINST library.
4. Run CSQ8SLDQ.</td></tr>
</table>

Table 51. One of the products has been updated to a new release in a new SMP/E environment and libraries (continued)		
Product	**Action if using CALLLIBS and SMP/E V3r2 or later** **Note: You do not need to run three separate jobs for Language Environment and Callable services. One job will suffice for both products.**	**Action if using LINK**
Callable services	1. Change the DDDEF for CSSLIB to point to the new library 2. Set the Boundary on your SMP/E job to the Target zone. 3. On the SMPCNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See *SMP/E for z/OS: Commands* for further information. 4. Run the SMP/E job.	1. Delete the XZMOD subentries for the following LMOD entries in the IBM MQ for z/OS target zone: CMQXRCTL, CMQXSUPR, CSQBSRV, CSQILPLM, CSQXJST, CSQXRCTL, CSQXSUPR, CSQ3AMGP, CSQ3EPX, CSQ3REPL 2. Set up the appropriate ZONEINDEXs between the IBM MQ zones and the Callable Services zones. 3. Tailor CSQ8SLDQ to refer to the new zone on the FROMZONE parameter of the LINK commands. CSQ8SLDQ can be found in the SCSQINST library. 4. Run CSQ8SLDQ.

For an example of a job to relink modules when using CALLLIBS, see "Running a LINK CALLLIBS job" on page 626.

Running a LINK CALLLIBS job

An example job to relink modules when using CALLLIBS.

The following is an example of the job to relink modules when using CALLLIBs on a SMP/E V3r2 system. You must provide a JOBCARD and the data set name of SMP/E CSI that contains IBM MQ for z/OS.

```
//*******************************************************************
//* RUN LINK CALLLIBS.
//*******************************************************************
//CALLLIBS EXEC PGM=GIMSMP,REGION=4096K
//SMPCSI  DD DSN=your.csi
//      DISP=SHR
//SYSPRINT DD SYSOUT=*
//SMPCNTL DD *
SET BDY(TZONE).
LINK LMODS CALLLIBS .
/*
```

Figure 112. Example SMP/E LINK CALLLIBS job

Using OTMA exits in IMS

Use this topic if you want to use IMS Open Transaction Manager Access exits with IBM MQ for z/OS.

If you want to send output from an IMS transaction to IBM MQ, and that transaction did not originate in IBM MQ, you need to code one or more IMS OTMA exits.

Similarly if you want to send output to a non-OTMA destination, and the transaction did originate in IBM MQ, you also need to code one or more IMS OTMA exits.

The following exits are available in IMS to enable you to customize processing between IMS and IBM MQ:

- An OTMA pre-routing exit
- A destination resolution user (DRU) exit

OTMA exit names

You must name the pre-routing exit DFSYPRX0. You can name the DRU exit anything, as long as it does not conflict with a module name already in IMS.

Specifying the destination resolution user exit name

You can use the *Druexit* parameter of the OTMACON keyword of the CSQ6SYSP macro to specify the name of the OTMA DRU exit to be run by IMS.

To simplify object identification, consider adopting a naming convention of DRU0xxxx, where xxxx is the name of your IBM MQ queue manager.

If you do not specify the name of a DRU exit in the OTMACON parameter, the default is DFSYDRU0. A sample of this module is supplied by IMS. See the *IMS/ESA® Customization Guide* for information about this.

Naming convention for IMS destination

You need a naming convention for the destination to which you send the output from your IMS program. This is the destination that is set in the CHNG call of your IMS application, or that is preset in the IMS PSB.

A sample scenario for an OTMA exit

Use the following topics for an example of a pre-routing exit and a destination resolution routing exit for IMS:

- “The pre-routing exit DFSYPRX0” on page 627
- “The destination resolution user exit” on page 629

To simplify identification, make the OTMA destination name similar to the IBM MQ queue manager name, for example the IBM MQ queue manager name repeated. In this case, if the IBM MQ queue manager name is " **VCPE** ", the destination set by the CHNG call is " **VCPEVCPE** ".

Related concepts

IBM MQ and IMS

“Using IBM MQ with IMS” on page 616

The IBM MQ -IMS adapter, and the IBM MQ - IMS bridge are the two components which allow IBM MQ to interact with IMS.

IMS and IMS bridge applications on IBM MQ for z/OS

The pre-routing exit DFSYPRX0

This topic contains a sample pre-routing exit for OTMA in IMS.

You must first code a pre-routing exit DFSYPRX0. Parameters passed to this routine by IMS are documented in *IMS/ESA Customization Guide*.

This exit tests whether the message is intended for a known OTMA destination (in our example VCPEVCPE). If it is, the exit must check whether the transaction sending the message originated in OTMA. If the message originated in OTMA, it will have an OTMA header, so you should exit from DFSYPRX0 with register 15 set to zero.

- If the transaction sending the message did not originate in OTMA, you must set the client name to be a valid OTMA client. This is the XCF member-name of the IBM MQ queue manager to which you want to send the message. The *IMS/ESA Customization Guide* tells you where to set this value. We suggest you set your client name (in the OTMACON parameter of the CSQ6SYSP macro) is set to the queue manager name. This is the default. You should then exit from DFSYPRX0 setting register 15 to 4.

- If the transaction sending the message originated in OTMA, and the destination is non-OTMA, you should set register 15 to 8 and exit.
- In all other cases, you should set register 15 to zero.

If you set the OTMA client name to one that is not known to IMS, your application CHNG or ISRT call returns an A1 status code.

For an IMS system communicating with more than one IBM MQ queue manager, you should repeat the logic for each IBM MQ queue manager.

Sample assembler code is shown in Figure 113 on page 628:

```
TITLE 'DFSYPRX0: OTMA PRE-ROUTING USER EXIT'
DFSYPRX0 CSECT
DFSYPRX0 AMODE 31
DFSYPRX0 RMODE ANY
*
SAVE (14,12),,DFSYPRX0&SYSDATE&SYSTIME
SPACE 2
LR  R12,R15        MODULE ADDRESSABILITY
USING DFSYPRX0,R12
*
L   R2,12(,R1)      R2 -> OTMA PREROUTE PARMS
*
LA  R3,48(,R2)      R3 AT ORIGINAL OTMA CLIENT (IF ANY)
CLC  0(16,R3),=XL16'00'  OTMA ORIG?
BNE  OTMAIN        YES, GO TO THAT CODE
*
NOOTMAIN DS 0H             NOT OTMA INPUT
LA  R5,8(,R2)       R5 IS AT THE DESTINATION NAME
CLC  0(8,R5),=C'VCPEVCPE' IS IT THE OTMA UNSOLICITED DEST?
BNE  EXIT0         NO, NORMAL PROCESSING
*
L   R4,80(,R2)      R4 AT ADDR OF OTMA CLIENT
MVC  0(16,R4),=CL16'VCPE' CLIENT OVERRIDE
B   EXIT4         AND EXIT
*
OTMAIN  DS 0H            OTMA INPUT
LA  R5,8(,R2)       R5 IS AT THE DESTINATION NAME
CLC  0(8,R5),=C'VCPEVCPE' IS IT THE OTMA UNSOLICITED DEST?
BNE  EXIT8         NO, NORMAL PROCESSING
```

```
*
EXIT0  DS 0H
LA  R15,0         RC = 0
B   BYEBYE
*
EXIT4  DS 0H
LA  R15,4         RC = 4
B   BYEBYE
*
EXIT8  DS 0H
LA  R15,8         RC = 8
B   BYEBYE
*
BYEBYE  DS 0H
RETURN (14,12),,RC=(15)   RETURN WITH RETURN CODE IN R15
SPACE 2
REQUATE
SPACE 2
END
```

Figure 113. OTMA pre-routing exit assembler sample

The destination resolution user exit

This topic contains a sample destination resolution user exit for IMS.

If you have set registers 15 to 4 in DFSYPRX0, or if the source of the transaction was OTMA ***and*** you set Register 15 to zero, your DRU exit is invoked. In this example, the DRU exit name is DRU0VCPE.

The DRU exit checks if the destination is VCPEVCPE. If it is, it sets the OTMA user data (in the OTMA prefix) as follows:

Offset
: **OTMA user data**

(decimal)

0
: OTMA user data length (in this example, 334)

2
: MQMD

326
: Reply to format

These offsets are where the IBM MQ - IMS bridge expects to find this information.

We suggest that the DRU exit is as simple as possible. Therefore, in this sample, all messages originating in IMS for a particular IBM MQ queue manager are put to the same IBM MQ queue.

If the message needs to be persistent, IMS must use a synchronized transaction pipe. To do this, the DRU exit must set the OUTPUT flag. For further details, refer to the *IMS/ESA Customization Guide*.

Write an IBM MQ application to process this queue, and use information from the MQMD structure, the MQIIH structure (if present), or the user data, to route each message to its destination.

A sample assembler DRU exit is shown in Figure 114 on page 630.

```
TITLE 'DRU0VCPE: OTMA DESTINATION RESOLUTION USER EXIT'
DRU0VCPE CSECT
DRU0VCPE AMODE 31
DRU0VCPE RMODE ANY
*
SAVE (14,12),,DRU0VCPE&SYSDATE&SYSTIME
SPACE 2
LR  R12,R15             MODULE ADDRESSABILITY
USING DRU0VCPE,R12
*
L   R2,12(,R1)            R2 -> OTMA DRU PARMS
*
L   R5,88(,R2)            R5 ADDR OF OTMA USERDATA
LA  R6,2(,R5)            R6 ADDR OF MQMD
USING MQMD,R6             AS A BASE
*
LA  R4,MQMD_LENGTH+10       SET THE OTMA USERDATA LEN
STH  R4,0(,R5)            = LL + MQMD + 8
*                        CLEAR REST OF USERDATA
MVI  0(R6),X'00'          ...NULL FIRST BYTE
MVC  1(255,R6),0(R6)        ...AND PROPAGATE IT
MVC  256(MQMD_LENGTH-256+8,R6),255(R6) ...AND PROPAGATE IT
*
VCPE   DS  0H
CLC  44(16,R2),=CL16'VCPE'     IS DESTINATION VCPE?
BNE  EXIT4             NO, THEN DEST IS NON-OTMA
MVC  MQMD_REPLYTOQ,=CL48'IMS.BRIDGE.UNSOLICITED.QUEUE'
MVC  MQMD_REPLYTOQMGR,=CL48'VCPE'  SET QNAME AND QMGRNAME
MVC  MQMD_FORMAT,MQFMT_IMS     SET MQMD FORMAT NAME
MVC  MQMD_LENGTH(8,R6),MQFMT_IMS_VAR_STRING
*                        SET REPLYTO FORMAT NAME
B   EXIT0
*
EXIT0  DS  0H
LA  R15,0             SET RC TO OTMA PROCESS
B   BYEBYE             AND EXIT
*
EXIT4  DS  0H
LA  R15,4             SET RC TO NON-OTMA
B   BYEBYE             AND EXIT
*
BYEBYE  DS  0H
RETURN (14,12),,RC=(15)       RETURN CODE IN R15
SPACE 2
REQUATE
SPACE 2
CMQA  EQUONLY=NO
CMQMDA DSECT=YES
SPACE 2
END
```

Figure 114. Sample assembler DRU exit

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document 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 document. The furnishing of this document does not grant 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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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 publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication 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 Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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 Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of WebSphere MQ.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, ibm.com®, are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Part Number:

(1P) P/N: