

10.0

Configuring IBM MQ



Note

Before using this information and the product it supports, read the information in [“Notices” on page 1115](#).

This edition applies to version 9 release 4 of IBM® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 2007, 2026.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Configuring.....	7
Creating queue managers on Multiplatforms.....	7
Configurable ephemeral directory.....	11
Userdata directory.....	12
Creating a default queue manager.....	12
Making an existing queue manager the default.....	14
Backing up configuration files after creating a queue manager.....	15
Moving a queue manager to a different operating system.....	15
Configuring connections between the client and server.....	16
Which communication type to use.....	17
How to set up an IBM MQ MQI client.....	19
Configuring an extended transactional client.....	20
Defining MQI channels.....	30
Creating and using AMQP channels.....	31
Creating server-connection and client-connection definitions on different platforms.....	36
Creating server-connection and client-connection definitions on the server.....	41
Channel-exit programs for MQI channels.....	58
Connecting a client to a queue sharing group.....	63
Using IBM MQ environment variables.....	63
Environment variables descriptions.....	65
Changing IBM MQ configuration information in .ini files on Multiplatforms.....	94
IBM MQ configuration file, mqs.ini.....	96
Queue manager configuration files, qm.ini.....	108
Installation configuration file, mqinst.ini	171
IBM MQ MQI client configuration file, mqclient.ini.....	172
Activity trace configuration file, mqat.ini.....	209
Configuring distributed queuing.....	212
IBM MQ distributed queuing techniques.....	213
Introduction to distributed queue management.....	233
Monitoring and controlling channels on AIX, Linux, and Windows.....	264
Monitoring and controlling channels on IBM i.....	287
Configuring a queue manager cluster.....	309
Configuring a uniform cluster.....	423
Configuring publish/subscribe messaging.....	444
Setting queued publish/subscribe message attributes.....	445
Starting queued publish/subscribe.....	446
Stopping queued publish/subscribe.....	446
Adding a stream.....	447
Deleting a stream.....	448
Adding a subscription point.....	448
Configuring distributed publish/subscribe networks.....	449
Configuring multiple installations.....	466
Connecting applications in a multiple installation environment.....	467
Changing the primary installation.....	474
Associating a queue manager with an installation.....	475
Finding installations of IBM MQ on a system.....	476
Configuring high availability and disaster recovery.....	477
Native HA.....	480
Comparison of Native HA CRR and Native HA IRR solutions.....	498
Native HA CRR.....	499
Native HA IRR.....	522
Multi-instance queue managers.....	540

External High Availability solutions.....	600
RDQM high availability.....	622
RDQM disaster recovery.....	658
RDQM disaster recovery and high availability.....	685
Combining highly available queue managers with IBM MQ clusters.....	714
Configuring recovery and restart.....	715
Automatic client reconnection.....	716
Console message monitoring.....	722
Logging: Making sure that messages are not lost.....	726
Backing up and restoring IBM MQ queue manager data.....	759
Changes to cluster error recovery on servers on Multiplatforms.....	766
Configuring JMS and Jakarta Messaging resources.....	767
Configuring connection factories and destinations in a JNDI namespace.....	768
Configuring JMS 2.0 objects using IBM MQ Explorer.....	772
Configuring JMS and Jakarta Messaging objects using the administration tools.....	773
Configuring JMS 2.0 resources in WebSphere Application Server.....	784
Configuring WebSphere Application Server to use the latest resource adapter maintenance level.....	793
Configuring the JMS PROVIDERVERSION property.....	796
Removing WebSphere Application Server durable subscriptions.....	803
Configuring Managed File Transfer.....	805
MFT configuration options on Multiplatforms.....	806
MFT configuration options on z/OS.....	808
Downloading and configuring Redistributable Managed File Transfer components.....	808
Creating an MFT Agent or Logger command data set.....	814
Configuring Managed File Transfer for z/OS.....	815
Configuring MFT on IBM i.....	846
Configuring MFT for first use.....	848
Configuring MFT agent queue managers.....	858
Configuring an MFT logger.....	868
Configuring the Connect:Direct bridge.....	901
Configuring the IBM MQ Console and REST API.....	906
Basic configuration for the mqweb server.....	907
Configuring the stand-alone IBM MQ Web Server.....	911
Configuring security.....	912
Configuring the HTTP host name.....	912
Configuring the HTTP and HTTPS ports.....	913
Configuring the response timeout.....	915
Configuring autostart.....	916
Configuring logging.....	917
Configuring the LTPA token.....	920
Configuring remote queue manager connection behavior for the IBM MQ Console.....	922
Configuring the administrative REST API gateway.....	924
Configuring the messaging REST API.....	925
Configuring the REST API for MFT.....	931
Tuning the mqweb server JVM.....	937
File structure of the IBM MQ Console and REST API installation component.....	938
Backing up and restoring your mqweb server configuration.....	940
Defining an Aspera gateway connection on Linux or Windows platforms.....	942
Configuring IBM MQ for use with the IBM Cloud Private metering service.....	947
Configuring a queue manager for use with the metering service instance on IBM Cloud Private... ..	948
Connecting to the IBM Cloud Private metering service through an HTTP proxy.....	951
Troubleshooting the connection to the metering service.....	951
Configuring queue managers on z/OS.....	952
Preparing to customize queue managers on z/OS.....	952
Setting up IBM MQ for z/OS.....	957
Testing a queue manager on z/OS.....	1021
Setting up communications with other queue managers on z/OS.....	1030

Using IBM MQ with IMS.....	1059
Using IBM MQ with CICS.....	1067
Upgrading and applying service to Language Environment or z/OS Callable Services.....	1067
Using OTMA exits in IMS.....	1069
Enabling MFT agents to connect to remote z/OS queue managers.....	1073
Configuring IBM MQ Internet Pass-Thru.....	1074
HTTP support in MQIPT.....	1074
SOCKS support in MQIPT.....	1076
SSL/TLS support in MQIPT.....	1077
Security exits in MQIPT.....	1104
Port number control in MQIPT.....	1108
Encrypting stored passwords in MQIPT.....	1108
Other security considerations for MQIPT.....	1110
Connection logs in MQIPT.....	1111
Configuring IBM MQ Internet Pass-Thru using containers.....	1112
Configuring streaming queues	1112
Notices.....	1115
Programming interface information.....	1116
Trademarks.....	1116

Configuring IBM MQ

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.

About this task

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](#).

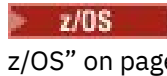
For instructions on how to create, start, stop, and delete a queue manager, see [“Creating queue managers on Multiplatforms”](#) on page 7.

For information about how to create the components required to connect your IBM MQ installations and applications together, see [“Configuring distributed queuing”](#) on page 212.

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 16.

For instructions on how to configure a queue manager cluster, see [“Configuring a queue manager cluster”](#) on page 309.

You can change the behavior of IBM MQ or a queue manager by changing configuration information. For more information, see [“Changing IBM MQ configuration information in .ini files on Multiplatforms”](#) on page 94. 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.

 For instructions on how to configure IBM MQ for z/OS®, see [“Configuring queue managers on z/OS”](#) on page 952.

Related concepts

[IBM MQ technical overview](#)

Related tasks

[Administering local IBM MQ objects](#)

[Administering remote IBM MQ objects](#)

 [Administering IBMi](#)

 [Administering IBM MQ for z/OS](#)

[Planning](#)

 [Planning your IBM MQ environment on z/OS](#)

[“Configuring queue managers on z/OS”](#) on page 952

Use these instructions to configure queue managers on IBM MQ for z/OS.

 Multiplatforms

Creating queue managers on Multiplatforms

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

Before you begin

Important: 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 that you must consider, especially in a production environment. Work through the following checklist:

The installation associated with the queue manager

To create a queue manager, you use the IBM MQ control command **crtmqm**. 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 that the Windows installer does not add the user that performs the installation to the mqm group, for more details, see [Authority to administer IBM MQ on AIX®, Linux®, and Windows](#).

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 shown 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.

Consider the coded character set identifier

The queue manager has an associated coded character set identifier (CCSID), which specifies the code page that is used for character or string data within the queue manager. This data includes configuration data, such as object names and descriptions, and any data that enters or leaves the queue manager (for example, through the MQI and other messaging APIs) where that data does not explicitly identify the code page to use. For example, the code page for character data within a message body is explicitly identified by the CCSID field in the message descriptor (MQMD). However, character data within the MQMD is assumed to be in the CCSID of the local queue manager (though this can be converted, where possible, at the entry or exit interfaces, for example during an MQGET call).

You might need to consider the CCSID, for example if you have a distributed messaging system with queue managers on different operating systems or in different countries. The CCSID is selected automatically when you create the queue manager, depending on the platform and locale of the user that runs the **crtmqm** command:

-   On Windows and all Linux platforms, the CCSID is the first CCSID in the `ccsid_part2.tbl` file that matches the character set of the current locale.

- **AIX** On AIX, character conversion is handled by the operating system, and system libraries are also queried for the appropriate CCSID based on the locale at queue manager creation. For example, you can use the `export LANG=C` command to set the locale to C, which uses the codeset ISO8859-1 and sets the queue manager's CCSID to 819. The locales that are available depend on the AIX filesets that are installed on the system. For example, if you want to convert data to and from Unicode, install the filesets that are listed in the "Unicode support on AIX" section of [Hardware and software requirements on AIX systems](#).
- **IBM i** On IBM i, the queue manager obtains its CCSID value from the job that created the queue manager. If that job has a special CCSID value of 65535, the default job CCSID is used, which is derived from the LANGID of the job. For example, LANGID(ENU) results in a default CCSID of 37. To determine the effective CCSID value that will be used when you create the queue manager, enter `DSPJOB OPTION(*DFNA)` then page through the output to see the Coded character set identifier, Language Identifier, and Default coded character set identifier values. To change either the CCSID or LANGID values, use the `CHGJOB` command.

You can identify the character set for the current locale by using the `locale charmap` command on AIX and Linux systems, and the `chcp` command on Windows. To show the CCSID of the queue manager after it is created, use the `DISPLAY QMGR MQSC` command.

If you need to specify a different CCSID, you can do so after you create the queue manager, by using the `ALTER QMGR MQSC` command, the `MQCMD_CHANGE_Q_MGR` PCF command, or (on IBM i systems) the `CHGMQM CL` command.

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 manager 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 12.

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 [Displaying and altering queue manager attributes](#) 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 [“Logging: Making sure that messages are not lost”](#) on page 726.

AIX

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.

About this task

To create a queue manager, you use the IBM MQ control command **crtmqm**. For more information, see **crtmqm**. The **crtmqm** command automatically creates the required default objects and system objects (see System default objects). Default objects form the basis of any object definitions that you make; system objects are required for queue manager operation.

Windows

On Windows systems you have the option to start multiple instances of the queue manager by using the **sax** option of the **crtmqm** command.

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

Procedure

- For information to help you with creating and managing queue managers, see the following subtopics:
 - [“Creating a default queue manager”](#) on page 12

- [“Making an existing queue manager the default” on page 14](#)
- [“Backing up configuration files after creating a queue manager” on page 15](#)

Related concepts

[Working with queue managers](#)

Related tasks

[Creating a queue manager called QM1](#)

[“Changing IBM MQ configuration information in .ini files on Multiplatforms” on page 94](#)

You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation by editing the information in the configuration (.ini) files. You can also change configuration options for IBM MQ MQI clients.

[“Configuring queue managers on z/OS” on page 952](#)

Use these instructions to configure queue managers on IBM MQ for z/OS.

Related reference

[System and default objects](#)

[crtmqm](#)

Configurable ephemeral directory

The configurable ephemeral directory defines the location that data ephemeral to the queue manager should go. This can be used to allow the AIX and Linux domain sockets to be placed on a non-mounted file system in a Red Hat® OpenShift® environment.

Prior to IBM MQ 9.2.0, on the AIX and Linux platforms, when a queue manager is running, AIX and Linux domain sockets are created under the `/var/mqm/sockets` directory. When running the queue manager inside a container, with `/var/mqm` as a mounted file system, some Linux platforms can prevent the creation of these domain sockets, because they allow some processes from outside the container to interfere with operations inside the container. This issue prevents IBM MQ from running in a Red Hat OpenShift container platform, under the default security context.

From IBM MQ 9.2.0, the **EphemeralPrefix** attribute can be used to configure the location of the ephemeral directory. If you do not use this attribute, you will see no change in behavior.

When a queue manager entry is created in `mqm.ini` (either using the **crtmqm** or **addmqinf** commands) the **EphemeralPrefix** attribute is added if you:

- Set the **DefaultEphemeralPrefix** attribute in the [“AllQueueManagers stanza of the mqm.ini file” on page 100](#).
- Set the **MQ_EPHEMERAL_PREFIX** environment variable.
- Specify **-v EphemeralPrefix** for the **addmqinf** command only.

You can also explicitly add the **EphemeralPrefix** attribute to an existing queue manager when it is stopped, and this is added when the queue manager is restarted.

If you specify an **EphemeralPrefix** attribute, when the queue manager starts it causes data ephemeral to the queue manager to be created under that prefix rather than its usual location. That is:

- Socket files that are usually present under `/var/mqm/sockets/<QM>` will now be under `/EphemeralPrefix/sockets/<QM>`
- Subpool files that are usually present under `/Prefix/qmgs/<QM>/@<Subpool>` will now be under `/EphemeralPrefix/qmgs/<QM>/@<Subpool>`

Notes:

- `/var/mqm/sockets/@SYSTEM` remains in its fixed location and is not part of the **EphemeralPrefix** attribute.
- `AMQCLCHL.TAB` remains under the `/Prefix/qmgs/<QM>/@ipcc` and is not part of the **EphemeralPrefix** attribute.

- The default **EphemeralPrefix** for a queue manager if one isn't specified is the same as the Prefix for the queue manager, but is also configurable. For more information, see [QueueManager stanza of the mqs.ini file](#).

The number of characters that the **EphemeralPrefix** attribute can include depends on your platform:

-   On AIX and Linux platforms it is limited to 12 characters.
-  On IBM i it is limited to 24 characters.

If you specify an **EphemeralPrefix** attribute that is too long, or does not exist, you receive an [AMQ7001E](#) message:

AMQ7001E: The location specified for the queue manager is not valid

Userdata directory

You can use the `userdata` directory to store persistent application status.

Each IBM MQ queue manager has a dedicated filesystem for its persistent state, which includes both its queue data and recovery log. The filesystem includes a `userdata` directory that you can use to store persistent status information for your applications. See [Directory content on Unix and Linux Systems](#) and [Directory content on Windows systems](#).

The `userdata` directory can be useful in a number of situations, for example:

- In RDQM configurations so that application information also moves when a queue manager fails over to another node (see [“Storing persistent application status”](#) on page 645).
- For multi-instance queue managers so their application state is located with their queue manager data on the shared network filesystem.
- More generally, where applications are configured queue manager services.

If you choose to store application state in the `userdata` directory, you must be aware that data written to this location might consume the available disk space allocated to the queue manager. You must ensure that sufficient disk space remains available for the queue manager to write queue data, logs, and other persistent state information.

The `userdata` directory has `mqm` user and group ownership, and it is world readable so users can access it without needing to be in the IBM MQ administrator group (that is, `mqm`). You cannot modify the permissions of the `userdata` directory, but you can create content in it with whatever ownership and permissions you require.

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 queue managers on Multiplatforms”](#) on page 7.

-   When you use **crtmqm** to create a queue manager on AIX and Linux, 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

[Working with local queues](#)

Related tasks

[“Backing up configuration files after creating a queue manager” on page 15](#)

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

[Displaying and altering queue manager attributes](#)

Related reference

[System and default objects](#)

Making an existing queue manager the default

You can make an existing queue manager the default queue manager either manually by using a text editor or, on Windows and Linux, by using IBM MQ Explorer.

About this task

To use a text editor to make an existing queue manager the default queue manager, complete the following steps.

WindowsLinux

On Windows and Linux (x86 and x86-64 platforms) systems, if you prefer to use IBM MQ Explorer to make this change, see [“Using IBM MQ Explorer to make a queue manager the default” on page 14](#).

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 (`mq.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 `mq.ini`, replacing the unwanted default queue manager with that of the one you want.

Related tasks

[“Changing IBM MQ configuration information in .ini files on Multiplatforms” on page 94](#)

You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation by editing the information in the configuration (`.ini`) files. You can also change configuration options for IBM MQ MQI clients.

WindowsLinux

Using IBM MQ Explorer to make a queue manager the default

On Windows and Linux (x86 and x86-64 platforms) systems, you can use IBM MQ Explorer to make an existing queue manager the default queue manager.

About this task

To use IBM MQ Explorer to make an existing queue manager the default queue manager on Windows and Linux (x86 and x86-64 platforms) systems, complete the following steps.

If you prefer to use a text editor to make this change manually, see [“Making an existing queue manager the default” on page 14](#).

Procedure

1. Open IBM MQ Explorer.
2. Right-click **IBM 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**.

Backing up configuration files after creating a queue manager

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

About this task

As a general rule, back up your configuration files each time you create a new queue manager.

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.

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

-  On Windows systems: `amqp_win.properties`
-   On AIX and Linux systems: `amqp_unix.properties`

Related tasks

[“Changing IBM MQ configuration information in .ini files on Multiplatforms” on page 94](#)

You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation by editing the information in the configuration (`.ini`) files. You can also change configuration options for IBM MQ MQI clients.

[“Backing up and restoring IBM MQ queue manager data” on page 759](#)

You can protect queue managers against possible corruption caused by hardware failures by backing up queue managers and queue manager data, by backing up the queue manager configuration only, and by using a backup queue manager.

Moving a queue manager to a different operating system

Follow these instructions to move a queue manager from one operating system to another. Note that this is **not** a migration of a queue manager.

About this task

You move a queue manager by re-creating it on the target system. The procedure re-creates the configuration of the queue manager, it does not attempt to re-create the current state of the queue manager by, for example, unloading and reloading queues.

Procedure

1. Log in to the source system as a user in the IBM MQ administrators (`mqm`) group.
2. Save the configuration information of the queue manager that you want to move by typing the following command:

```
dmpmqcfg -a -m QM_name > QM_file
```

Where:

- `QM_name` is the name of the queue manager that you want to move.
- `QM_file` is the name and path of a local file on the source system that the configuration information is written to.

See **dmpmqc.fg** for more information.

3. If the queue manager is part of a distributed configuration, quiesce the queue manager. Ensure that there are no messages in flight, then stop the queue manager.
4. If you are moving from one version of the product to another, migrate the queue manager on your current operating system, to the latest version.

If your current operating system is:

-  **Windows** Windows, see Migrating a queue manager on Windows
-  **Linux**  **AIX** AIX or Linux, see Migrating a queue manager on AIX and Linux

You should ensure that your existing applications still work.

5. Create an empty queue manager on your new operating system, using **crtmqm**.
6. Copy your object definitions across to the new queue manager you have just created, using **dmpmqc.fg**.

You should take great care when you copy the object definitions, because some manual modification of the definitions might be needed:

- Various attributes need to be checked and might need to be changed. These include:
 - IP addresses and ports in channels, listeners, and other objects
 - Security information such as user IDs
 - **startcmd** in services
 - Various other attributes.
- Durable subscribers that are not administered might lose messages.
- Other queue managers might also need to be changed so that their channels connect to the moved queue manager.

Once the definitions have been copied across, the application messages from the queue manager, on your original operating system, need to be copied to the queue manager on your new operating system, using an application that moves messages. Then check your applications still work.

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.

About this task

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.

Before you define your MQI channels, you must decide on which form of communication you are going to use and define the connection at each end of the channel.

If you are defining an MQI channel between an IBM MQ MQI client and a queue manager that are on different physical networks, or that communicate through a firewall, using IBM MQ Internet Pass-Thru might simplify the configuration. For more information, see IBM MQ Internet Pass-Thru.

Procedure

1. Decide on the form of communication that you are going to use.
See “Which communication type to use” on page 17.
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*.

Related concepts

[“IBM MQ MQI client configuration file, mqclient.ini” on page 172](#)

You can 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

[“Using IBM MQ environment variables” on page 63](#)

You can use commands to display the current settings or to reset the values of IBM MQ environment variables.

[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.

Types of transmission protocol for MQI channels

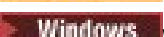
Depending on your client and server platforms, there are up to four types of transmission protocol for MQI channels:

- TCP/IP
- LU 6.2
- NetBIOS
- SPX

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.

Table 1. Transmission protocols - combination of IBM MQ MQI client and server platforms		
Transmission protocol	IBM MQ MQI client	IBM MQ server
TCP/IP “1” on page 18	 IBM i  AIX  Linux  Windows	 IBM i  AIX  Linux  Windows  z/OS

Table 1. Transmission protocols - combination of IBM MQ MQI client and server platforms (continued)		
Transmission protocol	IBM MQ MQI client	IBM MQ server
LU 6.2	 AIX  Linux ^{“2” on page 18}  Windows	 IBM i  AIX  Linux ^{“2” on page 18}  Windows  z/OS
NetBIOS	 Windows	 Windows
SPX	 Windows	 Windows

Notes:

1.   A message channel that uses TCP/IP can be pointed at an IBM Aspera[®] faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. See [Defining an Aspera gateway connection on Linux or Windows](#).
2. Except Linux (POWER platform)

Related concepts

[“Defining a TCP connection on Windows” on page 275](#)

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 AIX and Linux” on page 282](#)

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 302](#)

You can define a TCP connection within the channel definition using the Connection Name field.

[“Defining a TCP connection on z/OS” on page 1050](#)

To define a TCP connection, there are a number of settings to configure.

[“Defining an LU 6.2 connection on Windows” on page 277](#)

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 AIX and Linux” on page 286](#)

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 303](#)

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 278](#)

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

Related tasks

[“Defining an Aspera gateway connection on Linux or Windows platforms” on page 942](#)

The IBM Aspera faspio Gateway provides a fast TCP/IP tunnel that can significantly increase network throughput for IBM MQ. A queue manager running on any entitled platform can connect through an Aspera gateway. The gateway itself is deployed on Red Hat or Ubuntu Linux, or Windows.

Related reference

[“TCP/IP connection limits” on page 19](#)

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 1053](#)

To define an LU6.2 connection there are a number of settings to configure.

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 2. Maximum outstanding connection requests queued at a TCP/IP port	
Server platform	Maximum connection requests
 AIX	100
 Linux	100
 IBM i	255
 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 AIX 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.

How to set up an IBM MQ MQI client

Follow these instructions to set up a client.

Before you begin

To set up an IBM MQ MQI client you must have an IBM MQ server already installed and working, to which your client will connect.

Procedure

1. Check that you have a suitable platform for an IBM MQ MQI client and that the hardware and software satisfy the requirements.

Platform support is described in [Platform support for IBM MQ clients](#).

2. Decide how you are going to install IBM MQ on your client workstation, and then follow the instructions for your particular combination of client and server platforms.

Installation is described in the following topics:

-  [Installing an IBM MQ client on AIX](#)
-  [Installing an IBM MQ client on Linux](#)
-  [Installing an IBM MQ client on Windows](#)
-  [Installing an IBM MQ client on IBM i](#)

3. Ensure that your communication links are configured and connected.

Configuration of communication links is described in [Configuring connections between the server and client](#).

4. Check that your installation is working correctly.

See the verification section of the installation procedure for the platform, or platforms, that your enterprise use.

5. When you have the verified IBM MQ MQI client installation, consider whether you must secure your client.

Client security is described in [Setting up IBM MQ MQI client security](#).

6. Set up the channels between the IBM MQ MQI client and server that are required by the IBM MQ applications you want to run on the client.

Setting up channels is described in [Defining MQI channels](#). There are some additional considerations if you are using TLS.

These considerations are described in [Specifying that an MQI channel uses TLS](#). You might need to use an IBM MQ MQI client configuration file or IBM MQ environment variables to set up the channels. IBM MQ environment variables are described in [Using IBM MQ environment variables](#).

7. See [Developing applications](#) for a full description of IBM MQ applications.

8. When designing, building, and running applications in the IBM MQ MQI client environment, you need to consider the differences from a queue manager environment.

For information about these differences, see:

- [Using the message queue interface \(MQI\) in a client application](#)
- [Building applications for IBM MQ MQI clients](#)
- [Connecting IBM MQ MQI client applications to queue managers](#)
- [Resolving problems with IBM MQ MQI clients](#)

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

WebSphere Application Server 6 and later includes an IBM MQ messaging provider, so you do not need to use the extended transactional client.

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 on AIX](#)
-  [Installing an IBM MQ client on Linux](#)
-  [Installing an IBM MQ client on Windows](#)
-  [Installing an IBM MQ client on IBM i](#)

Using the information for your platform, 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 23](#).

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 26](#).

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 28](#)
- [“Configuring an extended transactional client for Tuxedo” on page 29](#)

Related concepts

[“The CHANNEL, TRPTYPE, CONNAME, and QMNAME parameters of the xa_open string” on page 25](#)
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 26](#)
The xa_open call fails in certain circumstances.

Related tasks

[“Using the extended transactional client with TLS channels” on page 27](#)

You cannot set up an TLS 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 26](#)

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 26](#)

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 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 MQ for z/OS.

Where an incompatible sequence is detected, IBM 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.

Another example is that xa_end receives abend 5C6-00D40079.

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

Ensure that you have enabled changes to XA client connections on IBM MQ for z/OS which allow the transaction manager to prepare a transaction on a different connection.

Notes:

- The change is not enabled by default. To make use of the change you must specify the keyword CSQSERVICE1 (in upper case) anywhere in the description field of the SVRCONN channel used by the XA client.
- 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. The disposition is determined by the name given on the xa_open call. If the queue sharing group name is used, then the XA connection requests a group unit of recovery.
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.

The change was enabled at IBM MQ for z/OS 9.0 through [APAR P173140](#)

Related concepts

[“Configuring XA-compliant transaction managers” on page 21](#)

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 protocols 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. Multiple hosts and ports for a queue manager may be specified by using a semicolon separator, for example:

```
host1(1415);host2(1416);host3(1417)
```

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 queue sharing group 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. 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 the transaction manager's connection. The user ID and password populate the MQCSP object on the MQCONN 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.

➤ **z/OS** If the application uses a queue sharing group 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 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 3 on page 26](#) for the assumed values of the AXLIB parameter.

Table 3. 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	 Windows systems	libEncServer
Tuxedo	 AIX	/usr/lpp/tuxedo/lib/libtux.a(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. Here 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 4 on page 27](#).

Table 4. IBM MQ libraries containing the XA switch structures	
Platform	Library containing the XA switch structures
 AIX  Linux	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 27](#)

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 TLS channels

You cannot set up an TLS 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 TLS 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 TLS 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 TLS 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
-  Windows

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 5 on page 28](#) for the fully qualified path name of each switch load file.

Table 5. The switch load files	
Platform	Switch load file
 AIX  Linux	MQ_INSTALLATION_PATH/lib/amqczsc
Windows	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,connname=MARS(1415),qmname=MARS,tpm=cics" \
  SwitchLoadFile="C:\Program Files\IBM\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 [Securing](#).

- The CICS task termination exits that are supplied for use on IBM MQ client systems are listed in [Table 6 on page 29](#). 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 6. CICS task termination exits

Platform	Source	Library
 AIX  Linux	amqzscgx.c	amqczscg
 Windows 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, the name of the file is `ubbstxcx.cfg`.

 On Windows, the name of the file is `ubbstxcn.cfg`.

- In the entry for a queue manager in the Tuxedo resource manager table, specify the name of an XA switch structure and the fully qualified path name of the library that contains the structure:

–

 On AIX, specify `udataobj/RM`.

–

 On Windows, specify `udataobj\rm`.

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 Microsoft Transaction Server (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 MQCONNEX 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 MQCONNEX 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. Here 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 36](#) 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 41](#) for details on how to use this method.

Automatically defined channels



IBM MQ products on Multiplatforms 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 242](#)

The channel control function provides facilities for you to define, monitor, and control channels.

ALW Creating and using AMQP channels

When you install the IBM MQ support for the AMQP Service Component 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](#)
- ▶ **Linux** [IBM MQ rpm components for Linux systems](#)
- ▶ **Linux** [IBM MQ Debian components for Linux Ubuntu systems](#)
- ▶ **Windows** [IBM MQ features for Windows systems](#)

Note: See [Restarting the IBM MQ service for AMQP](#) for an example of a SERVICE component and more information if your AMQP service stops working correctly.

This task also assumes that you have an existing queue manager.

To make a test connection to the queue manager, you can use any AMQP clients that implement the OASIS AMQP 1.0 protocol, such as MQ Light and Apache Qpid clients like Apache Qpid Proton and Apache Qpid JMS.

From IBM MQ 9.3.0, you can use only the default channel, SYSTEM.DEF.AMQP, to test MQ Light connections to the queue manager. The following procedure uses the default channel.

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.

Note: AMQP channels do not support user defined AMQP services. AMQP channels only support the system default SYSTEM.AMQP.SERVICE service. You can only define one instance of this service for each queue manager.

Important: From IBM MQ 9.4.0, CMS key repositories and stash files are not supported with AMQP and MQTT channels that use SSL/TLS. Use PKCS #12 key repositories and protect the key repository passwords by using the IBM MQ password protection system instead. For more information, see [Setting up a key repository on AIX, Linux, and Windows](#).

Procedure

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

```
runmqsc QMNAME
```

2. (Only necessary if your queue manager is from IBM MQ 9.4.0 or if it is IBM MQ 9.0.4 or earlier.) 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)
```

Note: From IBM MQ 9.4.0 the SYSTEM.AMQP.SERVICE has its **CONTROL** attribute set to MANUAL. This prevents the service from being started when the queue manager is started. Setting the **CONTROL** property to QMGR, starts automatically when the queue manager is started.

From IBM MQ 9.1 to IBM MQ 9.3, the SYSTEM.AMQP.SERVICE has its **CONTROL** attribute set to QMGR.

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 the **SSLCAUTH** attribute. 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)
```

- 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 receive 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](#)

▶ ALW 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 the AMQP service component 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](#)

▶ ALW 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.
- amqp_*.json, 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, amqp_0.json 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.

JSON formatted logs in AMQP are optional and need to be enabled manually. You do so by modifying the [AMQP logs, error logs, and configuration files](#).

Related concepts

[Error logs on AIX, Linux, and Windows](#)

Related tasks

[Developing AMQP client applications](#)

[Securing AMQP clients](#)

[Enabling JSON formatted logs for AMQP](#)

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

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

About this task

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.

 On z/OS you can also use the Operation and Control panels.

 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.

The following considerations apply when **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**.

Procedure

- To define a server-connection channel on the server, see [“Defining a server-connection channel on the server” on page 36](#).
- To create a client-connection channel on an IBM MQ MQI client by using the **MQSERVER** environment variable, see [“Creating a client-connection channel on the IBM MQ MQI client using MQSERVER” on page 37](#).
- To create a client-connection channel on an IBM MQ MQI client by using the MQCNO structure on an MQCONN call, see [“Creating a client-connection channel on the IBM MQ MQI client using MQCNO” on page 41](#).

Defining a server-connection channel on the server

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

Procedure

1. Optional: If you are using a Multiplatforms server, 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 TLS Subject Distinguished Name. For more information, see [Channel authentication records](#)

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

You can define a client-connection channel on a client workstation by using the **MQSERVER** environment variable.

About this task

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.

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).

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 AIX and Linux, 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 the **MQCHLLIB** and **MQCHLTAB** environment variables.

Procedure

- Depending on your platform, use one of the following commands to specify the channel definition with **MQSERVER**.

–  On Windows, specify a simple channel definition as follows:

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

For example:

```
export MQSERVER='SYSTEM.DEF.SVRCONN/TCP/AMACHINE.ACOMPANY.COM(1414) '
```

-   On AIX and Linux, specify a simple channel definition as follows:

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

For example:

```
SET MQSERVER=SYSTEM.DEF.SVRCONN/TCP/AMACHINE.ACOMPANY.COM(1414)
```

-  On IBM i, specify a simple channel definition as follows:

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

For example:

```
ADDENVVAR ENVVAR(MQSERVER) VALUE('SYSTEM.DEF.SVRCONN/TCP/AMACHINE.ACOMPANY.COM(1414)')
```

Notes:

- The *ChannelName* must be the same name as defined on the server. It 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.
- The *TransportType* can be one of LU62, TCP, NETBIOS, SPX, depending on your IBM MQ client platform.
-   On AIX and Linux, the *TransportType* is case-sensitive and must be uppercase. An MQCONN or MQCONNX call returns 2058 if the transport type is not recognized
- The *ConnectionName* is the name of the server as defined to the communications protocol (*TransportType*). It must be a fully-qualified network name, for example AMACHINE.ACOMPANY.COM(1414).
- The *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 connection name 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 connection name list to specify different queue manager instances.
- To cancel **MQSERVER** and return to the client channel definition table pointed to by **MQCHLLIB** and **MQCHLTAB**, enter the following command:

-   On AIX and Linux:

```
unset MQSERVER
```

-  On Windows:

```
SET MQSERVER=
```

Example

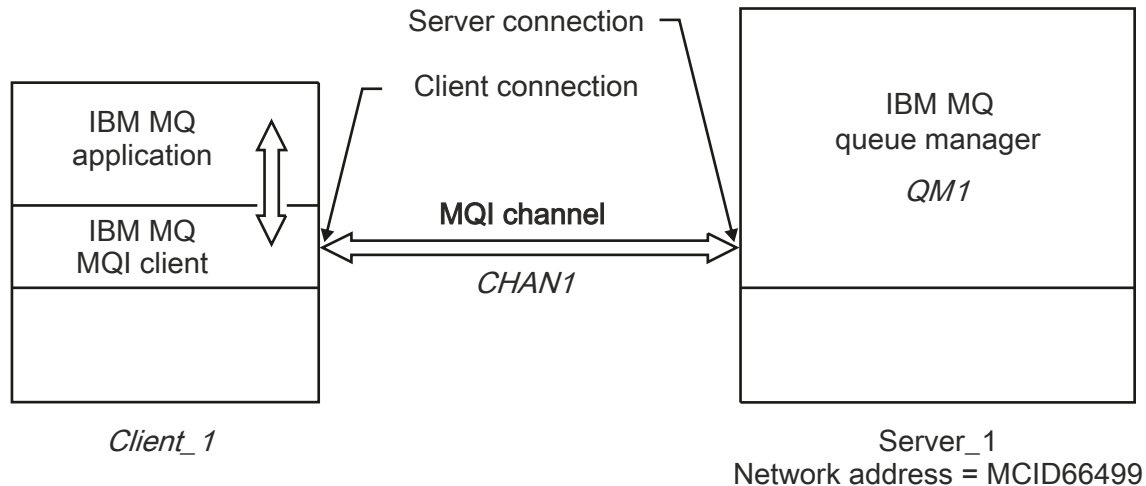


Figure 1. Example of a simple channel definition

To create the simple channel definition shown in [Figure 1 on page 39](#), use the following commands:

- **Linux** **AIX** On AIX and Linux:

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

- **Windows** On Windows:

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

Note: For information on how to change the TCP/IP port number, see [“Changing the TCP/IP default port” on page 40](#).

Some more examples of simple channel definitions are as follows:

- **Windows** On Windows:

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

- **Linux** **AIX** On AIX and Linux:

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

where BOX99 is the LU 6.2 ConnectionName.

- **IBM i** On IBM i:

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

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**.

Related tasks

[“Using IBM MQ environment variables” on page 63](#)

You can use commands to display the current settings or to reset the values of IBM MQ environment variables.

[“Creating a client-connection channel on the IBM MQ MQI client using MQCNO” on page 41](#)

You can define a client-connection channel on the client workstation by using the MQCNO structure on an MQCONN call.

Changing the TCP/IP default port

By default, for TCP/IP, IBM MQ assumes that the channel will be connected to port 1414. If required, you can change the default value.

About this task

You can change the port number by using one of the following three options:

- By using the **MQSERVER** environment variable.
- By changing the `mqclient.ini` file.
- By adding IBM MQ to the services file.

Procedure

- To change the port number by using the **MQSERVER** environment variable, add the port number in brackets as the last part of the *ConnectionName*, for example:

–   On AIX and Linux:

```
export MQSERVER='ChannelName/TransportType/ConnectionName(PortNumber)'
```

–  On Windows:

```
SET MQSERVER=ChannelName/TransportType/ConnectionName(PortNumber)
```

- To change the port number by using the `mq.ini` file, add the port number to the protocol name, for example:

```
TCP:
port=2001
```

- To change the port number by adding IBM MQ to the services file, complete the steps described in [“Using the TCP/IP listener on AIX and Linux” on page 283](#).

Changing the SPX default socket

By default, for SPX, IBM MQ assumes that the channel will be connected to socket 5E86. If required, you can change the default value.

About this task

You can change the port number by using either of the following options:

- By using the **MQSERVER** environment variable.

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.

- By changing the SPX stanza of the `mqclient.ini` file. Changing the `qm.ini` file.

Procedure

- To change the port number for an SPX connection by using the **MQSERVER** environment variable, specify the *ConnectionName* and socket in the form `network.node(socket)` as shown in the following example:

```
SET MQSERVER=ChannelName/TransportType/ConnectionName(SocketNumber)
```

Note: If the IBM MQ client and server are on the same network, you do not need to specify the network. If you are using the default socket, you do not need to specify the socket.

- To change the port number by using the `qm.ini` file, add the port number to the protocol name, for example:

```
SPX:  
socket=5E87
```

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

You can define a client-connection channel on the client workstation by using the MQCNO structure on an MQCONN call.

About this task

An IBM MQ MQI client application can use the connect options structure, MQCNO, on an **MQCONN** 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 Transport Layer Security (TLS), a client application can also provide information relating to TLS in the MQCD structure. Additional information relating to TLS can be provided in the TLS configuration options structure, MQSCO, which is also referenced by the MQCNO structure on an **MQCONN** call.

For more information about the MQCNO, MQCD, and MQSCO structures, see [MQCNO](#), [MQCD](#), and [MQSCO](#).

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

Related tasks

[“Creating a client-connection channel on the IBM MQ MQI client using MQSERVER” on page 37](#)

You can define a client-connection channel on a client workstation by using the **MQSERVER** environment variable.

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.

About this task

You 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 to define a server-connection channel on the server machine.
-   On Linux and Windows, you can also use IBM MQ Explorer.
-  On z/OS, you can also use the Operation and Control panels.
-  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).

Procedure

1. To define a server-connection channel, see [“Defining the server-connection channel on the server” on page 55](#).
2. To define a client-connection channel, see [“Defining the client-connection channel on the server” on page 56](#).

Related tasks

[“Configuring a binary format CCDT” on page 43](#)

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On Multiplatforms, a binary CCDT containing default settings is created automatically when the queue manager is created. You use the **runmqsc** command to update a binary CCDT.

[“Defining the server-connection channel on the server” on page 55](#)

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

[“Defining the client-connection channel on the server” on page 56](#)

After you have defined the server-connection channel, you can define the corresponding client-connection channel.

[“Accessing client-connection channel definitions” on page 57](#)

You can 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. You can also locate a client channel definition table (CCDT) through a URL.

Configuring client channel definition tables

A client channel definition table (CCDT) defines client connection channels and their attributes. Clients read this file to determine which queue managers to connect to. The CCDT file can be either JSON or binary format.

About this task

The queue manager does not read the CCDT file. It is only used to provide channel definitions and authentication information to clients.

A binary format CCDT is created automatically when a queue manager is created. You update the client channel definitions stored in this table using only the **runmqsc** command.

A JSON format CCDT is a flat text file with a .json extension. You create and update this table manually, which is less restrictive than using the **runmqsc** command.

 z/OS JMS clients running within an application server use a CCDT to reference remote queue manager connection details. From IBM MQ for z/OS 9.1, IBM MQ Advanced for z/OS Value Unit Edition allows JMS clients to remotely connect to queue managers on other z/OS LPARs. Therefore these clients can also use CCDTs.

To help you configure CCDTs to work with your clients, choose from the following tasks:

Procedure

- [“Configuring a binary format CCDT” on page 43](#)
- [“Configuring a JSON format CCDT” on page 45](#)
- [“Locations for the CCDT” on page 52](#)
- [“URL access to the CCDT” on page 53](#)

Related concepts

MQI client: [Client Channel Definition Table \(CCDT\)](#)

Related tasks

[“Configuring a uniform cluster” on page 423](#)

Uniform clusters permit applications to be designed for scale and availability, and can connect to any of the queue managers within that uniform cluster.

Configuring a binary format CCDT

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On Multiplatforms, a binary CCDT containing default settings is created automatically when the queue manager is created. You use the **runmqsc** command to update a binary CCDT.

Before you begin

you can also create a CCDT in JavaScript Object Notation (JSON) format, and using this alternative format has some advantages over using a binary CCDT. See [“Configuring a JSON format CCDT” on page 45](#).

Clients on all platforms can view and use CCDTs. However the binary CCDT can only be created and modified under IBM MQ for Multiplatforms.

About this task

 On [Multiplatforms](#):

- A binary CCDT is created automatically in the @ipcc directory under the data directory for the queue manager.
- As well as being created automatically, the binary CCDT associated with a queue manager is kept in sync with the object definitions. When you define, alter or delete a client channel object, both the queue manager object definition and the entry in the CCDT are updated as part of the same operation.

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 use the **runmqsc** command to change the location and contents of the binary CCDT.

Clients on all platforms can view and use a binary CCDT.

Procedure

Multiplatforms

Create a default binary CCDT.

On [Multiplatforms](#), a default binary 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:

-  On IBM i, in the integrated file system:

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

-   On AIX and Linux systems:

`/prefix/qmgrs/QUEUEMANAGERNAME/@ipcc`

The name of the directory referenced by `QUEUEMANAGERNAME` is case sensitive on AIX 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.

-  On Windows:

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

where `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.

- Locate the CCDT:
 - On the client computer
 - At a location shared by more than one client
 - On the server as a shared file

See “[Locations for the CCDT](#)” on page 52.

a) Create a binary CCDT directly on a client machine.

- Use the `runmqsc` command with the `-n` parameter.
- The CCDT is created in the location indicated by `MQCHLLIB`, and with the filename indicated by `MQCHLTAB`, which is `AMQCLCHL . TAB` by default.
- **Important:** If you specify the `-n` parameter, you must not specify any other parameter.

b) Change the location.

You can change the path to the CCDT by setting `MQCHLLIB`. Bear in mind that, if you have multiple queue managers on the same server, they share the same CCDT location.

- Access the CCDT

You can access the CCDT:

- Remotely from a URL, by defining the `MQCCDTURL` environment variable.
- Locally by setting the `MQCHLLIB` and `MQCHLTAB` environment variables.
- Locally by defining the `ChannelDefinitionDirectory` and `ChannelDefinitionFile` attributes of the `CHANNELS` stanza in the client configuration file.

See “[Locations for the CCDT](#)” on page 52 for various examples.

- View or edit the CCDT contents.

You can view the CCDT contents with the `runmqsc` command:

1. Set the environment variables to [Access the CCDT](#)
2. Run the command `runmqsc -n`
3. Run the command `DISPLAY CHANNEL(*)`, for example

 On [Multiplatforms](#), you can also edit the binary CCDT contents using the `runmqsc` command. 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. See [runmqsc](#) for more examples of using the command.

- Provide clients with the authentication information to check for TLS certificate revocation.
 - a) Define a namelist containing authentication information objects.
 - b) Set the queue manager attribute `SSLCRLNL` to the name of the namelist.

Related concepts

[Working with revoked certificates](#)

Related tasks

[“Configuring a JSON format CCDT” on page 45](#)

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. You use a text editor to create and update a JavaScript Object Notation (JSON) CCDT.

Configuring a JSON format CCDT

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. You use a text editor to create and update a JavaScript Object Notation (JSON) CCDT.

Before you begin

 If you are using IBM MQ for Multiplatforms, you can instead use the binary CCDT that is created automatically when you create a queue manager. See [“Configuring a binary format CCDT” on page 43](#).

About this task

The file name of the CCDT schema for the JSON format is:

Linux

`/opt/mqm/lib/ccdt_schema.json`

Windows

`C:\Program Files\IBM\MQ\bin\ccdt_schema.json`

There is no default JSON CCDT, and IBM MQ does not supply any tooling to create or edit CCDTs in JSON format. Nonetheless, you have more configuration options when you manually develop a JSON CCDT than when you use the **runmqsc** command to work with a binary CCDT:

- You do not need to be using IBM MQ for Multiplatforms to create and edit a JSON CCDT file.
- Using the JSON format, you can define duplicate channel definitions of the same name. When you deploy IBM MQ on the cloud, you can use this to make your deployment scalable and highly available.
- The JSON file is human readable, which can simplify queue manager configuration.
- A flat file format can be integrated with:
 - Version control tooling to track the CCDT history
 - Automation tooling in continuous delivery
- You need no specialist tooling to maintain the CCDT file.
- The file is smaller.
- This format provides backwards and forwards compatibility.

Notes:

1. The JSON standard sees duplicate keys as valid, however, the JSON parser only takes the last read value of duplicate keys when assigning attributes. Therefore, when defining duplicate channels, each channel must be an element of an array value that is assigned to the 'channel' key.
2. The key and attribute values in JSON CCDTs **must** be provided exactly as specified in the schema. Especially note that the JSON parsing is case-sensitive.
3. JSON CCDTs do not support storing of Lightweight Directory Access Protocol (LDAP) server locations for Certificate Revocation Lists (CRL) and Online Certificate Status Protocol (OCSP) responder location information.
4. If a key-value pair within a user exit entry (in the `exits` section) does not have a value, that key-value pair is ignored.

Table 7. Encoding requirements by platform

Platform	JMS client encoding	C client encoding
 IBM i	ASCII	EBCDIC
 AIX, Linux, and Windows	ASCII	ASCII
 z/OS	Either ASCII or EBCDIC	Not applicable



Attention: When you provide any definition for a channel through a JSON CCDT (including a *sparse* one which does not include all attributes), a complete channel definition is constructed with all attributes defined, using the defaults for anything not specified in the JSON.

Therefore, you must provide specific values for each attribute for which you do not want the default value.

Procedure

- Create a JSON CCDT
 - a) Create a flat file with a .json extension with a generic text editor.
 - b) Define a CCDT.

See “JSON CCDT examples” on page 49 and “Channel attributes supported by the JSON CCDT” on page 47.

- Locate the CCDT:
 - On the client computer
 - At a location shared by more than one client
 - On the server as a shared file

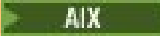
See “Locations for the CCDT” on page 52.

- Validate a JSON CCDT

Validate the CCDT against the schema with a JSON linter.

See [How to validate an IBM MQ CCDT JSON file against the schema](#) for information on how to create a CCDT file with two channels and validate that it works.

The CCDT Schema is included with the product and client packages:

-   On AIX and Linux systems:
\$MQ_INSTALLATION_PATH/lib and /lib in the product and client packages respectively.
-  On Windows:
%MQ_INSTALLATION_PATH%\bin and \bin in the product and client packages respectively.

Notes:

- JSON linters are available online.
- The schema defines mandatory attributes with the 'required' key.
- The schema defines attribute data types with the 'type' key.
- Access the CCDT

You can access the CCDT:

 - Remotely from a URL, by defining the **MQCCDTURL** environment variable.
 - Locally by setting the **MQCHLLIB** and **MQCHLTAB** environment variables.

- Locally by defining the **ChannelDefinitionDirectory** and **ChannelDefinitionFile** attributes of the CHANNELS stanza in the client configuration file.

See “Locations for the CCDT” on page 52 for various examples.

- View or edit the CCDT contents

Each entry of a CCDT represents a client connection to a specific queue manager. You can view or edit the CCDT contents with a text editor.

If you want only to view the CCDT, you can also do this using the **runmqsc** command as follows:

1. Set environment variables to give you access to the CCDT, as described in the previous step.
2. Run the command `runmqsc -n` . For more information, see [runmqsc](#).
3. Run the **DISPLAY CHANNEL** command. For example, run `DISPLAY CHANNEL(*)` .

Related concepts

[Working with revoked certificates](#)

Related tasks

[“Configuring a binary format CCDT” on page 43](#)

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On Multiplatforms, a binary CCDT containing default settings is created automatically when the queue manager is created. You use the **runmqsc** command to update a binary CCDT.

[“Configuring a uniform cluster” on page 423](#)

Uniform clusters permit applications to be designed for scale and availability, and can connect to any of the queue managers within that uniform cluster.

Channel attributes supported by the JSON CCDT

A list of the client connection channel attributes supported by the JSON CCDT. This list is a subset of the attributes supported by the binary CCDT.

Attribute mapping

These attributes are inserted into the following channel object:

```
{ "channel": [ { $CHANNEL_1_KEY_VALUE_LIST }, ..., { $CHANNEL_N_KEY_VALUE_LIST } ] }
```

where `$CHANNEL_X_KEY_VALUE_LIST` is a comma separated list of the attributes listed in the following table.

See “JSON CCDT examples” on page 49 for basic use cases.

The JSON schema is shipped in `/opt/mqm/lib/ccdt_schema.json`. To discover what values are valid for each of the attributes look at the JSON schema.

The following table lists the JSON object, key, and data type, together with the corresponding binary channel attribute definition.



Attention: The required attributes are the channel **name** and channel **type**. If you also define **portRange**, the *low* and *high* attributes are also required.

JSON object	JSON key	JSON data type	Binary attribute definition
channel (array)	name	STRING	CHANNEL
channel (array)	type	STRING	CHLTYPE
channel.clientConnection	queueManager	STRING	QMNAME
channel.clientConnection.connection (array)	host	STRING	CONNNAME
channel.clientConnection.connection	port	INT	CONNNAME

JSON object	JSON key	JSON data type	Binary attribute definition
channel.compression.header (array)	header	STRING	COMPHDR
channel.compression.message (array)	message	STRING	COMPMSG
channel.connectionManagement	affinity	STRING	AFFINITY
channel.connectionManagement	clientWeight	INT	CLNTWGHT
channel.connectionManagement	defaultReconnect	STRING	DEFRECON
channel.connectionManagement	disconnectInterval	INT	DISCINT
channel.connectionManagement	heartbeatInterval	INT	HBINT
channel.connectionManagement	keepAliveInterval	INT	KAINT
channel.connectionManagement	sharingConversations	INT	SHARECNV
channel.connectionManagement.localAddress (array)	host	STRING	LOCLADDR
channel.connectionManagement.localAddress (array)	port	INT	LOCLADDR
channel.connectionManagement.localAddress.portRange	high	INT	LOCLADDR
channel.connectionManagement.localAddress.portRange	low	INT	LOCLADDR
channel.exits.receive (array)	name	STRING	RCVEXIT
channel.exits.receive (array)	userData	STRING	RCVDATA
channel.exits.security	name	STRING	SCYEXIT
channel.exits.security	userData	STRING	SCYDATA
channel.exits.send (array)	name	STRING	SENDEXIT
channel.exits.send (array)	userData	STRING	SENDATA
channel.general	description	STRING	DESCR
channel.general	maximumMessageLength	INT	MAXMSGL
channel.timestamps	altered	STRING	ALTDATE and ALTTIME
channel.transmissionSecurity	certificateLabel	STRING	CERTLABL
channel.transmissionSecurity	certificatePeerName	STRING	SSLPEER
channel.transmissionSecurity	cipherSpecification	STRING	SSLCIPH
 channel.transmissionSecurity	quantumSafeAlgorithm	STRING	SSLQS
 channel.transmissionSecurity	quantumSafeRequired	STRING	SSLQSR

Notes:

- channel.connectionManagement.localAddress can be defined as one of the following combinations of keys:
 - host and port

- host and portRange
- port
- portRange
- The `channel.timestamps` altered JSON key is optional and, if not defined, the value defaults to the JSON CCDT file last modified time. However, if the environment is configured to fetch the CCDT from a URL, the default value is the time the file was last downloaded.
- `channel.clientConnection.connection` must include both host and port keys.
- The altered key is a single string that encapsulates both ALTDATE and ALTTIME attributes.
- The transport type can only be TCP, therefore the following attributes are not defined in the schema:
 - **TRPTYPE**
 - **USERID**
 - **PASSWORD**
 - **MODENAME**
 - **TPNAME**

Related reference

[Channel attributes for channel types](#)

JSON CCDT examples

Use the examples listed in this topic as a basis for your requirements.

Open a generic text editor and copy one of the following examples:

- [“Define a simple client connection” on page 49](#)
- [“Define one channel and one queue manager using TLS” on page 49](#)
- [“Define one channel and one queue manager not using TLS” on page 50](#)
- [“Define two channels with the same name” on page 50](#)
- [“Complete list of CCDT channel attribute definitions for a client connection channel” on page 51](#)

Define a simple client connection

```
{
  "channel": [
    {
      "general": {
        "description": "a channel"
      },
      "name": "channel",
      "clientConnection": {
        "connection": [
          {
            "host": "localhost",
            "port": 1414
          }
        ],
        "queueManager": "QM1"
      },
      "type": "clientConnection"
    }
  ]
}
```

Define one channel and one queue manager using TLS

```
{
  "channel": [
```

```

{
  "name": "SSL.SVRCONN",
  "clientConnection": {
    "connection": [
      {
        "host": "aztlan1.fyre.ibm.com",
        "port": 1419
      }
    ],
    "queueManager": "QM92TLS"
  },
  "transmissionSecurity": {
    "cipherSpecification": "TLS_AES_128_GCM_SHA256",
    "certificateLabel": "ibmwebspheremqadministrator",
  },
  "type": "clientConnection"
}
]
}

```

Define one channel and one queue manager not using TLS

```

{
  "channel": [
    {
      "name": "SYSTEM.DEF.SVRCONN",
      "clientConnection": {
        "connection": [
          {
            "host": "aztlan1.fyre.ibm.com",
            "port": 1414
          }
        ],
        "queueManager": "QM92"
      },
      "type": "clientConnection"
    }
  ]
}

```

Define two channels with the same name

Each channel connects to two distinct queue managers:

```

{
  "channel":
  [
    {
      "general":
      {
        "description": "First channel"
      },
      "name": "channel",
      "clientConnection":
      {
        "connection":
        [
          {
            "host": "localhost",
            "port": 1414
          }
        ],
        "queueManager": "QM1"
      },
      "type": "clientConnection"
    },
    {
      "general":
      {
        "description": "Second channel"
      },
      "name": "channel",
      "clientConnection":
      {
        "connection":

```

```

    [
      {
        "host": "localhost",
        "port": 1415
      }
    ],
    "queueManager": "QM2"
  },
  "type": "clientConnection"
}
]
}

```

Complete list of CCDT channel attribute definitions for a client connection channel

Note: If a key-value pair within a user exit entry (in the exits section) does not have a value, that key-value pair is ignored.

```

{
  "channel":
  [
    {
      "compression":
      {
        "header": [ "system" ],
        "message": [ "zlibfast" ]
      },
      "connectionManagement":
      {
        "sharingConversations": 10,
        "clientWeight": 1,
        "affinity": "none",
        "defaultReconnect": "yes",
        "heartbeatInterval": 600,
        "keepAliveInterval": -1,
        "localAddress":
        [
          {
            "portRange":
            {
              "low": 2020,
              "high": 3030
            }
          }
        ]
      }
    }
  ],
  "exits":
  {
    "receive":
    [
      {
        "name": "",
        "userData": ""
      }
    ],
    "security":
    {
      "name": "",
      "userData": ""
    },
    "send":
    [
      {
        "name": "",
        "userData": ""
      }
    ]
  },
  "general":
  {
    "description": "First channel",
    "maximumMessageLength": 4194304
  },
  "name": "the_channel",
  "clientConnection":
  {
    "connection":
    [

```

```

        {
            "host": "localhost",
            "port": 1414
        }
    ],
    "queueManager": "QM1"
},
"timestamps":
{
    "altered": "2018-12-04T15:37:22.000Z"
},
"transmissionSecurity":
{
    "cipherSpecification": "",
    "certificateLabel": "",
    "certificatePeerName": ""
},
"type": "clientConnection"
}
]
}

```

Related reference

[Channel attributes for channel types](#)

[Channel attributes in alphabetical order](#)

Locations for the CCDT

IBM MQ supports retrieving a CCDT from a URL. You can make the CCDT accessible to the client as a shared file, while it remains located on the server. Alternatively you can distribute the CCDT, either by copying the CCDT to individual client computers or by copying the CCDT to a location that is shared by more than one client.

The URL can use the following protocols: file, ftp, or http. V 10.0.0 From IBM MQ 10.0.0, https URLs are also supported.

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.

V 10.0.0 If you use HTTP to copy the file, use HTTP over TLS (HTTPS) if possible, to protect the integrity of the CCDT file in transit. To use HTTPS, you must first configure an appropriate truststore for the client library. For more information, read about the [HTTPSKeyStore](#) attribute of the `mqclient.ini` file.

How to host the CCDT file on a server

The CCDT can be hosted in a central location that is accessible through a URL, removing the need to individually update the CCDT for each deployed client. Native (C/C++, COBOL and RPG) and unmanaged .NET applications can pull the CCDT from a URL.

The default caching behavior of IBM MQ clients is that a CCDT file is only pulled down if the file modification time is different from the last time that it was retrieved. As with most client configuration options, there are a variety of ways in which the URL location can be provided:

- [CCDTURLPtr](#) and [CCDTURLoffset](#) through the MQCNO structure being passed into MQCONNX MQI call
- [MQCCDTURL](#) environment variable
- [ChannelDefinitionDirectory](#) attribute in the Channels stanza of `mqclient.ini`

Both authenticated and unauthenticated URLs are supported. Here are some examples:

```
export MQCCDTURL=ftp://myuser:password@myhost.sample.com//var/mqm/qmgrs/QMGR/@ipcc/AMQCLCHL.TAB
```

```
export MQCCDTURL=http://myhost.sample.com/var/mqm/qmgrs/QMGR/@ipcc/AMQCLCHL.TAB
```

If you want to use this support with FTP or HTTP, you must still host the CCDT file on a server, but all of your client applications can automatically pick up changes to channel definitions without manually

pushing out updates or needing to mount a networked file system on each client. For more information, see [“URL access to the CCDT”](#) on page 53.

How to specify the location of the CCDT on the client

On a client system, you can specify the location of the CCDT in the following 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 **ChannelDefinitionDirectory** attribute to specify the directory where the table is located, and the **ChannelDefinitionFile** attribute to specify the file name.
- By providing a URL for a CCDT that is hosted in a central location, as previously described.

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.

If you use a URL to provide the location of the CCDT, the order of precedence for a native client application to find the client channel definition is as described in [“URL access to the CCDT”](#) on page 53.

URL access to the CCDT

You can host a client channel definition table (CCDT) in a central location that can be accessed through a URL, removing the need to individually update the CCDT for each deployed client.

A client channel definition table can be located through a URL in any of the following ways:

- By programming using MQCNO
- By using environment variables. The client channel definitions used to connect to the queue manager are located using the following environment variables in this order of precedence: **MQSERVER**, **MQCHLLIB**, and **MQCHLTAB**.



Attention: You can use the environment variable option to provide the URL only for native programs connecting as clients, that is C, COBOL, or C++ applications. The environment variables have no effect for Java, JMS or managed .NET applications.

- By using the `mqclient.ini` file Channels stanza.

IBM MQ supports retrieving a CCDT from a file, ftp, or http URL. V 10.0.0 From IBM MQ 10.0.0, https (HTTP over TLS) URLs are also supported. Use HTTPS URLs to protect the integrity of the CCDT file in transit. To use HTTPS, you must first configure an appropriate truststore to enable the validation of HTTP server certificates. For more information, read about the [HTTPSKeyStore](#) attribute of the `mqclient.ini` file.

V 10.0.0 You can enable FIPS compliant mode when securely retrieving a CCDT by enabling FIPS for TLS communications. To enable FIPS mode for TLS communication, see [Specifying that only FIPS certified CipherSpecs are used at run time on the MQI client](#).

You can use the **MQCCDTURL** environment variable to provide a URL as a single value from which a client channel definition table can be obtained.

You can also use the directory path specified by the **MQCHLLIB** environment variable (or the path specified by the **ChannelDefinitionDirectory** attribute in the “Channels stanza of the client configuration file” on page 190) to locate a CCDT file, either through a URL or in the local file system directory (for example, `/var/mqm`). Note that an **MQCHLLIB** value is a directory stem and works in combination with **MQCHLTAB** to derive the fully qualified URL.

V 10.0.0 From IBM MQ 10.0.0, you can supply a customized `SSLSocketFactory` instance to control how your IBM MQ classes for JMS/Jakarta Messaging application securely retrieves CCDT files from HTTPS endpoints. This customized `SSLSocketFactory` instance can also specify the level of validation that the IBM MQ classes for JMS/Jakarta Messaging library runs when it retrieves the file. For more information, see [Supplying a customized SSLSocketFactory in IBM MQ classes for JMS/Jakarta Messaging](#).

Basic authentication on connections is supported through the credentials being encoded in the URL:

Authenticated connections

```
export MQCHLLIB=ftp://myuser:password@myhost.sample.com/var/mqm/qmgrs/QMGR/@ipcc
export MQCHLLIB=http://myuser:password@myhost.sample.com/var/mqm/qmgrs/QMGR/@ipcc
```

Unauthenticated connections

```
export MQCHLLIB=ftp://myhost.sample.com/var/mqm/qmgrs/QMGR/@ipcc
export MQCHLLIB=http://myhost.sample.com/var/mqm/qmgrs/QMGR/@ipcc
export MQCHLLIB=file:///var/mqm/qmgrs/QMGR/@ipcc
```

Note: If you want to use authenticated connections you must, as with JMS, provide the user name and password encoded in the URL.

The order of precedence, for a native client application, to find a client channel definition is now:

1. MQCD provided by **ClientConnOffset** and **ClientConnPtr** in MQCNO.
2. URL provided by **CCDTUrlOffset** and **CCDTUrlPtr** in MQCNO.
3. **MQSERVER** environment variable.
4. If an `mqclient.ini` file is defined and the Channels stanza contains a **ServerConnectionParms** attribute, then the channel that it defines is used. For more information, see [“IBM MQ MQI client configuration file, mqclient.ini” on page 172](#) and [“Channels stanza of the client configuration file” on page 190](#).
5. **MQCCDTURL** environment variable.
6. **MQCHLLIB** and **MQCHLTAB** environment variable.
7. **ChannelDefinitionDirectory** and **ChannelDefinitionFile** in the [“Channels stanza of the client configuration file” on page 190](#).

Important: Access to a CCDT file using a URL always opens a read-only copy of the file, even when using the `file://` protocol.

Attempting to open a CCDT file for write access, for example when using the **DEFINE CHANNEL** MQSC command from a client, returns an error message indicating that the file could not be opened for write access.

It is, however, possible to read channel and authentication information definitions using **runmqsc**.

Related tasks

[“Accessing client-connection channel definitions” on page 57](#)

You can 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. You can also locate a client channel definition table (CCDT) through a URL.

[“Configuring a binary format CCDT” on page 43](#)

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On Multiplatforms, a binary CCDT containing default settings is created automatically when the queue manager is created. You use the **runmqsc** command to update a binary CCDT.

[Using a CCDT with IBM MQ classes for JMS](#)

Related reference

[CCDTURL](#)

[MQCNO - Connect options](#)

[XMSC_WMQ_CCDTURL](#)

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. [MQCONN](#) 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 TLS Subject Distinguished Name. For more information, see [Channel authentication records](#)

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

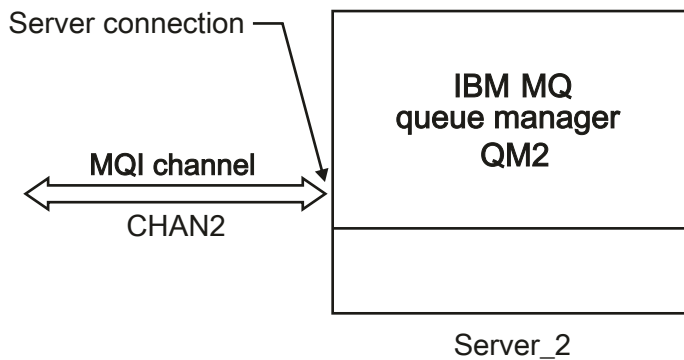


Figure 2. Defining the server-connection channel

Related tasks

“Defining the client-connection channel on the server” on page 56

After you have defined the server-connection channel, you can define the corresponding client-connection channel.

Defining the client-connection channel on the server

After you have defined the server-connection channel, you can define the corresponding client-connection channel.

Before you begin

Define the server-connection channel. For more information, see [“Defining the server-connection channel on the server” on page 55](#).

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 the **SET CHLAUTH** command 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 TLS Subject Distinguished Name. For more information, see [Channel authentication records](#)

Results

Multi On Multiplatforms, 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 [“Configuring a binary format CCDT”](#) on page 43.

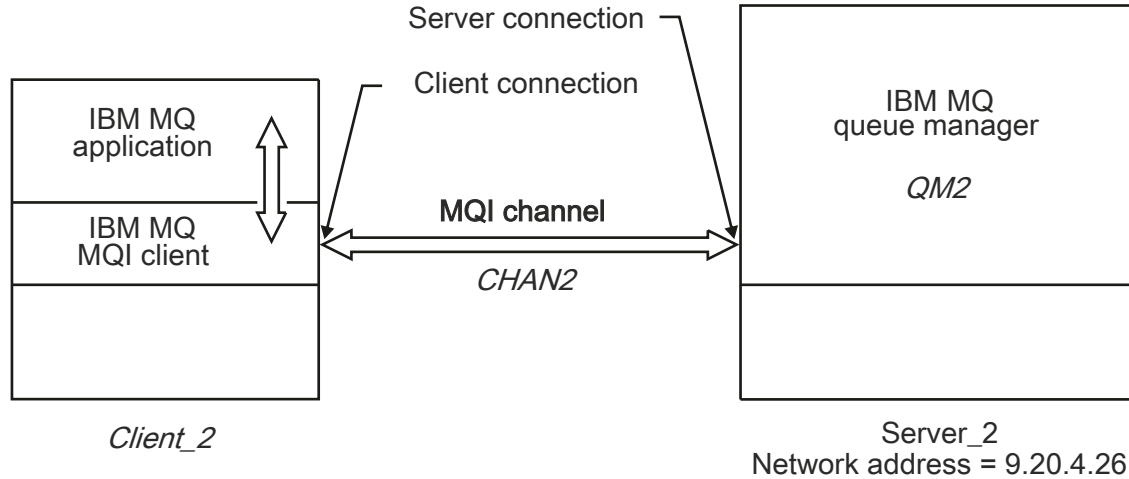


Figure 3. Defining the client-connection channel

Related reference

[DEFINE CHANNEL \(define a new channel\)](#)

[SET CHLAUTH \(create or modify a channel authentication record\)](#)

Accessing client-connection channel definitions

You can 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. You can also locate a client channel definition table (CCDT) through a URL.

Before you begin

This task assumes that you have defined, in a CCDT, the client-connection channels that you need. See [“Configuring client channel definition tables”](#) on page 42.

About this task

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

Procedure

1. Make the CCDT available to the client application, in one of the following 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.
 - d) Optional: Copy the CCDT to a central location from where it can be accessed through a URL by IBM MQ native (C/C++, COBOL and RPG) and unmanaged .NET applications.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 that contains the CCDT, in one of the following 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 190.
- b) Optional: Use the environment variables **MQCHLLIB** and **MQCHLTAB**.

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

-   On AIX and Linux systems:

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

-  On IBM i systems:

```
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.
- d) Provide the location of a centrally hosted CCDT through a URL, either by programming using MQCNO, using environment variables, or by using mqclient.ini file stanzas. For more information, see [“Locations for the CCDT”](#) on page 52 and [“URL access to the CCDT”](#) on page 53.

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.

Related tasks

[“Configuring a binary format CCDT”](#) on page 43

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On Multiplatforms, a binary CCDT containing default settings is created automatically when the queue manager is created. You use the **runmqsc** command to update a binary CCDT.

Related reference

[MQI client: Client Channel Definition Table \(CCDT\)](#)

Channel-exit programs for MQI channels

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

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 59](#)

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.

[User exits, API exits, and IBM MQ installable services](#)

Related tasks

[Extending queue manager facilities](#)

Related reference

[“Path to exits” on page 59](#)

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 61](#)

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.

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 AIX, 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 61](#) illustrates the use of security exits in a client connection, using the IBM MQ object authority manager to authenticate a user.

Either the SecurityParmsPtr or SecurityParmsOffset field in the MQCNO structure is set by the client and there are security exits at both ends of the channel. After the normal security message exchange ends, and the channel is ready to run, the MQCSP structure is passed to the client security exit. The exit can access the MQCSP structure by using the SecurityParms field in the MQCXP structure. The exit type is set to MQXR_SEC_PARMS. The security exit can alter the credentials in the MQCSP structure, or leave them unchanged.

The data that is 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. The exit can access the MQCSP structure by using the SecurityParms field in the MQCXP structure. The security exit receives and processes this data. This processing is typically to

reverse any change that is made to the credentials in the MQCSP structure by the client exit, which are then used to authorize the queue manager connection. The resulting MQCSP structure is referenced by using `SecurityParmsPtr` in the MQCNO structure on the queue manager system.

The memory address that is returned with the `SecurityParms` field of the MQCXP structure 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 the `SecurityParmsPtr` or `SecurityParmsOffset` field in the MQCNO structure is set, 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 the `SecurityParmsPtr` and `SecurityParmsOffset` fields in the MQCNO structure are not set, 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 that is addressed by the `SecurityParmsPtr` field. The security exit is not called again until it is terminated (ExitReason of MQXR_TERM). The exit writer can free the memory that is 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 WebSphere MQ 7.1 might require updating. For more information, see [Channel security exit programs](#).

ALW 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 62 , “2” on page 62	X'81'	X'91'
MQDISC “1” on page 62	X'82'	X'92'
MQOPEN “3” on page 62	X'83'	X'93'

Table 8. Identifying API calls (continued)

API call	Value of byte 10 for request	Value of byte 10 for reply
MQCLOSE	X'84'	X'94'
MQGET “4” on page 62	X'85'	X'95'
MQPUT “4” on page 62	X'86'	X'96'
MQPUT1 request “4” on page 62	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.

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.

About this task

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](#).

Procedure

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 must have the same definitions on each queue manager in the group.

1. Define the SVRCONN channel as shown in the following example:

```
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 MQCONNEX call.

2. Define the CLNTCONN channel as shown in the following example:

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

Results

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.

Using IBM MQ environment variables

You can use commands to display the current settings or to reset the values of IBM MQ environment variables.

About this task

You can use environment variables in the following ways:

- To set the variables in your system profile to make a permanent change

- To 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

For each environment variable, you can use commands to display the current setting or to reset the value of the environment variable. These commands are available on all supported platforms unless otherwise stated. The format of the command depends on your platform. For example:

-   On AIX and Linux:

```
export [environment variable]=value
```

-  On Windows:

```
Set [environment variable]=value
```

-  On IBM i:

```
ADDENVVAR ENVVAR(environment variable) VALUE(xx)
```

-  For IBM MQ Appliance, see [Configuring environment variables on the IBM MQ Appliance](#) in the IBM MQ Appliance documentation.

Where applicable, IBM MQ uses default values for those environment variables that you have not set.

Note:  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 IBM MQ environment variables on your client platform.

Procedure

- 

On Windows, for each environment variable, use the following commands to display the current setting or to reset the value of a variable:

- To remove the value of an environment variable, use the following command:

```
SET MQSERVER=
```

- To display the current setting of an environment variable, use the following command:

```
SET MQSERVER
```

- To display all environment variables for the session, use the following command:

```
set
```

-  

On AIX and Linux, for each environment variable, use the following commands to display the current setting or to reset the value of a variable:

- To remove the value of an environment variable, use the following command:

```
unset MQSERVER
```

- To display the current setting of an environment variable, use the following command:

```
echo $MQSERVER
```

- To display all environment variables for the session, use the following command:

set

Related tasks

[Setting environment variables for IBM MQ classes for JMS/Jakarta Messaging](#)

[Environment variables relevant to IBM MQ classes for Java](#)

[Defining additional environment variables in the service.env file](#)

[“Changing IBM MQ configuration information in .ini files on Multiplatforms” on page 94](#)

You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation by editing the information in the configuration (.ini) files. You can also change configuration options for IBM MQ MQI clients.

Related reference

[The use of environment variables in MFT properties](#)

Environment variables descriptions

Descriptions of server and client environment variables that are intended for customer use.

Examples of use

- ▶ **Linux** ▶ **AIX** On AIX and Linux systems, use this format: `export [environment variable]=value`.
- ▶ **Windows** On Windows systems, use this format: `Set [environment variable]=value`.
- ▶ **IBM i** On IBM i systems, use this format: `ADDENVVAR ENVVAR(environment variable) VALUE(xx)`.
- ▶ **MQ Appliance** For IBM MQ Appliance, see [Configuring environment variables on the IBM MQ Appliance](#) in the IBM MQ Appliance documentation.

Table 9. Summary of environment variables			
Name	Description	On server?	On client?
AMQ_ALLOWED_CIPHERS	Specifies a custom list of CipherSpecs that are enabled for use with IBM MQ channels.	✓	
▶ Multi V 3000 “AMQ_APP_INSTANCE_MODE” on page 72	Specifies whether all the connections that are created by one client Java virtual machine are treated as a single application instance.		✓
AMQ_BAD_COMMS_DATA_FDCS	Causes FFST files to be written for any bad data including simple known formats if IBM MQ receives data that is in an incorrect format from a host over TCP/IP.	✓	
AMQ_CONVEBCDICNEWLINE	Specifies how IBM MQ is to convert an EBCDIC NL character into ASCII format.	✓	
AMQ_DIAGNOSTIC_MESSAGE_SEVERITY	Specifies whether the message severity is to be appended to the message number when an IBM MQ process writes a message to an error log or to the console.	✓	

Table 9. Summary of environment variables (continued)

Name	Description	On server?	On client?
AMQ_DISABLE_CLIENT_AMS	For Java clients, if you want to connect to a queue manager from an earlier version of the product, disables IBM MQ Advanced Message Security (AMS) at the client.		✓
AMQ_DMPMQCFG_QSGDISP_DEFAULT	Specifies which types of definition to include when inquiring the disposition of a queue manager.	✓	
AMQ_IODELAY , AMQ_IODELAY_INMS and AMQ_IODELAY_FFST	Used to fine tune diagnostics and timings when the input and output for your queue manager log and storage file system is slow.	✓	
AMQ_LDAP_TRACE	Makes it possible to switch LDAP client trace on and off without also stopping or starting the queue manager.	✓	
AMQ_LICENSING_METRIC	Causes the queue manager to upload data relating to monthly VPC license types instead of the default behavior of uploading data related to hourly container-based licenses.	✓	
 Linux  AIX AMQ_MQS_INI_LOCATION	Specifies the location that is used for the <code>mqs.ini</code> file.	✓	
AMQ_NO_BAD_COMM_S_DATA_FDCS	Suppresses the generation of FFSTs when reporting error messages on the initial communications flow if the data that IBM MQ receives from a host over TCP/IP is in an incorrect format.	✓	
AMQ_NO_IPV6	Disables the use of IPv6 while attempting a connection.	✓	✓
“AMQ_RDQM_NO_PING_CHECKS” on page 75	Disables ICMP pings in an RDQM configuration.	✓	
AMQ_REVERSE_COMMIT_ORDER	Configures a queue manager so that in an XA transaction the IBM MQ queue manager change is committed after the corresponding database update is completed.	✓	
AMQ_SSL_ALLOW_DEFAULT_CERT	Allows the certificate that an application uses to connect to a queue manager to be a default certificate, provided that a default certificate is present in the key repository, and the key repository does not contain a personal certificate with the prefix <code>ibmwebspheremquserid</code> .	✓	

Table 9. Summary of environment variables (continued)

Name	Description	On server?	On client?
AMQ_SSL_LDAP_SERVER_VERSION	Specifies that either LDAP v2 or LDAP v3 is used by IBM MQ cryptographic components in cases where CRL servers require that a specific version of the LDAP protocol be used.	✓	
“AMQ_TLS_ENVIRONMENT_SCOPE” on page 76	Controls whether IBM MQ uses a single IBM Global Security Kit (GSKit) environment for the entire process or a IBM Global Security Kit (GSKit) environment per connection.		✓
 AMQ_USE_ZLIBNX	On AIX, enables message channel agents (MCA) to use the hardware-accelerated zlibNX library for compression and decompression of message data when using ZLIBFAST or ZLIBHIGH techniques.	✓	
GMQ_MQ_LIB	Specifies the client bindings library when IBM MQ automation classes for ActiveX (MQAX) are run against the client rather than the server.  Attention: This environment variable is removed at 9.2.		
 Linux  IBM i  AIX HOME	On AIX, Linux and IBM i, specifies the name of the directory that is searched for the <code>mqclient.ini</code> file.	✓	
 Windows HOMEDRIVE and HOMEPATH	On Windows, used together to specify the name of the directory that is searched for the <code>mqclient.ini</code> file.	✓	
LDAP_BASEDN	Required environment for running an LDAP sample program. Specifies the base Distinguished Name for the directory search.	✓	
LDAP_HOST	Optional for running an LDAP sample program. Specifies the name of the host where the LDAP server is running.	✓	
LDAP_VERSION	Optional for running an LDAP sample program. Specifies the version of the LDAP protocol to be used.	✓	
 MQCCDTCERTVPOL	An optional environment variable for CCDT certificate validation policy that can be configured for the IBM MQ classes for JMS client library.		✓

Table 9. Summary of environment variables (continued)

Name	Description	On server?	On client?
MQ_CHANNEL_SUPPRESS_INTERVAL	Specifies the time interval, in seconds, during which the messages defined with MQ_CHANNEL_SUPPRESS_MSGS are to be suppressed from being written to the error log, together with the number of times that a message will be allowed to occur during the specified time interval before being suppressed.	✓	
MQ_CHANNEL_SUPPRESS_MSGS	Suppresses channel error messages in the error log.	✓	
MQ_CONNECT_TYPE	Used in combination with the type of binding specified in the Options field of the MQCNO structure that is used on an MQCONN call.		✓
MQ_CROSS_QUEUE_ORDER_ALL	Specifies that the message put order is maintained in a unit of work.	✓	
MQ_EPHEMERAL_PREFIX	Specifies the path to the queue manager's ephemeral directory, within which ephemeral queue manager data is kept, while the queue manager is running.	✓	
 MQ_FILE_PATH	Location of the executable files. Set within the system environment variables during the first installation of IBM MQ. The same path is also stored within the Windows Registry.	✓	
MQ_JAVA_DATA_PATH	Specifies the directory for log and trace output for the IBM MQ classes for JMS and IBM MQ classes for Jakarta Messaging, and the IBM MQ classes for Java.	✓	
MQ_JAVA_INSTALL_PATH	Specifies the directory where the IBM MQ classes for JMS and IBM MQ classes for Jakarta Messaging, and the IBM MQ classes for Java are installed.	✓	
MQ_JAVA_LIB_PATH	Specifies the directory where the IBM MQ classes for JMS and IBM MQ classes for Jakarta Messaging, and the IBM MQ classes for Java libraries are stored.	✓	
 MQ_LCLADDR	On Multiplatforms, defines a default local address value that is used for all sender channels that do not have a local address defined.	✓	✓
“MQ_OVERRIDE_DATA_PATH” on page 79	Overrides the default directory of the IBM MQ data path.	✓	✓
 MQ_SET_NODELAYACK	On AIX, switches off TCP delayed acknowledgment.	✓	

Table 9. Summary of environment variables (continued)

Name	Description	On server?	On client?
“MQ_USER_NAME” on page 79	Allows an IBM MQ non-install image on Linux to choose the name of an unnamed user.	✓	
MQAPI_TRACE_LOGFILE	Defines the prefix for the user-specified file to which the sample API exit program generates an MQI trace.	✓	
 MQAPPLNAME	If the application name has not yet been chosen, specifies the name to be used to identify the connection to a queue manager.		✓
MQCCSID	Specifies the coded character set number to be used and overrides the CCSID value with which the server has been configured.		✓
MQCCDTURL	Provides the equivalent capability to setting a combination of the MQCHLLIB and MQCHLTAB environment variables.		✓
MQCERTLABL	Defines the certificate label of a channel definition for IBM MQ to use to locate a personal certificate that is sent during a TLS handshake.		✓
MQCERTVPOL	Specifies the type of certificate validation policy to be used.		✓
MQCHLLIB	Specifies the directory path to the file that contains the client channel definition table (CCDT).		✓
MQCHLTAB	Specifies the name of the file that contains the client channel definition table (CCDT).		✓
MQCLNTCF	Specifies the location of the IBM MQ MQI client configuration file.		✓
MQDOTNET_TRACE_ON	Enables trace for IBM MQ .NET redistributable clients.		✓
MQIPADDRV	Specifies which IP protocol to use for a channel connection.		✓
MQKEYRPWD	Specifies the password for the key repository that holds the digital certificate belonging to the user.		✓
 MQLICENSE	On Linux systems, used to accept or view an IBM MQ license after you install the product.	✓	✓
MQMAXERRORLOGSIZE	Specifies the size of the queue manager error log that is copied to the backup.	✓	
 MQNAME	Specifies the local NetBIOS name that the IBM MQ processes can use.	✓	✓

Table 9. Summary of environment variables (continued)

Name	Description	On server?	On client?
MQNOREMPOOL	Switches off channel pooling and causes channels to run as threads of the listener.	✓	
MQPSE_TRACE_LOGFILE	Describes where the trace files for the Publish Exit Sample Program must be written to.	✓	
MQS_AMSCRED_KEYFILE	Overrides or provides the initial key file to use at run time of IBM MQ Advanced Message Security (AMS) applications, or when you are protecting a keystore configuration file by using the runamscred command.		✓
MQS_DISABLE_ALL_INTERCEPT	For native C clients, if you want to connect to a queue manager from an earlier version of the product, disables IBM MQ Advanced Message Security (AMS) at the client.		✓
MQS_IPC_HOST	Sets the host name that is added to the directory path.	✓	
MQS_KEYSTORE_CONFIG	Specifies the location of the keystore configuration file for IBM MQ Advanced Message Security (AMS), if the file is not in the default location.		✓
MQS_MQI_KEYFILE	Specifies the location of an initial key file which contains the initial key to be used for password protection operations.		✓
MQS_SSLCRYP_KEYFILE	Specifies the full path and name of the file containing the initial key used to encrypt the password in the PKCS #11 cryptographic hardware configuration string.		✓
MQS_TRACE_OPTIONS	For selective component tracing on AIX, activates the high detail and parameter tracing functions individually.	✓	✓
 MQS_GETGROUPLIST_API	Returns all the groups information and looks for the requested mqm user in the groups, without causing an internal queue manager timeout.	✓	
MQSERVER	Defines a minimal channel, specifying the location of the IBM MQ server and the communication method to be used.		✓
MQSNOAUT	Disables the object authority manager (OAM) and prevents any security checking, for example in a test environment.	✓	
MQSPREFIX	As an alternative to changing the default prefix, overrides the DefaultPrefix for the crtmqm command.	✓	

Table 9. Summary of environment variables (continued)

Name	Description	On server?	On client?
 MQSSLCRYP	Holds a parameter string that you can use to configure the cryptographic hardware present on the system.		✓
MQSSLFIPS	 Specifies whether only FIPS certified cryptographic libraries and algorithms are to be used if cryptography is carried out in IBM MQ.		✓
MQSSLKEYR	Specifies the location of the key repository that holds the digital certificate belonging to the user.		✓ client and mqcertck tool
MQSSLPROXY	Specifies the host name and port number of the HTTP proxy server to be used by IBM Global Security Kit (GSKit) for OCSP checks.		✓
MQSSLRESET	Specifies the number of unencrypted bytes sent and received on a TLS channel before the TLS secret key is renegotiated.		✓
 MQSUITEB	Specifies whether Suite B compliant cryptography is to be used. In the instance that Suite B cryptography is used,		✓
MQTCPTIMEOUT	Specifies how long IBM MQ waits for a TCP connect call.	✓	✓
“MQTRACEPATH” on page 92	When taking long traces, sets the name of a directory that has more disk space.	✓	✓
ODQ_MSG	Sets the name of the file containing error and information messages, if you use a dead-letter queue handler that is different from runmqdlq .	✓	
ODQ_TRACE	Enables tracing if you use a dead-letter queue handler that is different from runmqdlq .	✓	
 TOKENCERTVPOL	An optional environment variable for JWT certificate validation policy that can be configured for the IBM MQ classes for JMS client library.		✓
WCF_TRACE_ON	Enables trace for the WCF custom channel.		✓
WMQSOAP_HOME	Used when completing additional configuration steps after the .NET SOAP over JMS service hosting environment is correctly installed and configured in IBM MQ.		✓

Table 9. Summary of environment variables (continued)

Name	Description	On server?	On client?
XMS_TRACE_ON, XMS_TRACE_FILE_PATH, XMS_TRACE_FORMAT, and XMS_TRACE_SPECIFICATION	Used to enable and configure XMS trace.		✓

AMQ_ALLOWED_CIPHERS

Multi

You can use the **AMQ_ALLOWED_CIPHERS** environment variable to specify a custom list of CipherSpecs that are enabled for use with IBM MQ channels on Multiplatforms. The environment variable takes the same values as the **AllowedCipherSpecs** SSL stanza attribute of the `.ini` file:

- A single CipherSpec name, or
- A comma-separated list of IBM MQ CipherSpec names to re-enable, or
- The special value of ALL, representing all CipherSpecs (not recommended).

Note: Enabling **ALL** CipherSpecs is not recommended as this will enable SSL 3.0 and TLS 1.0 protocols and a large number of weak cryptographic algorithms.

For more information, see [Providing a custom list of enabled CipherSpecs on Multiplatforms](#) in [CipherSpec](#) order in TLS handshake.

AMQ_APP_INSTANCE_MODE

Multi

V 10.0.0

When **AMQ_APP_INSTANCE_MODE** is set to JVM, all connections that are created by a single client Java process that is running in this environment are treated as a single application instance and connect or reconnect to the same queue manager. See [Application Instances in IBM MQ classes for JMS/Jakarta Messaging](#) for more information, including restrictions that apply to applications that enable this mode of operation.

AMQ_BAD_COMMS_DATA_FDCS

The **AMQ_BAD_COMMS_DATA_FDCS** environment variable is effective when set to any value.

If the data that IBM MQ receives from a host over TCP/IP is in an incorrect format, for example because a network client has connected to an IBM MQ listener port and attempted to communicate with an unsupported application protocol, the queue manager writes an AMQ9207E error message to the queue manager error logs. IBM MQ listeners support TCP/IP connections from queue manager message channel agents (MCAs) and from MQI, JMS and XMS client applications.

Note: IBM MQ listeners do not support the application protocol used by AMQP and MQTT clients, these clients should instead connect to the network ports configured in the applicable AMQP channel or MQXR telemetry service.

A failure data capture (FDC) record containing the invalid data that IBM MQ has received might also be written. However, an FFST file is not generated if this is the beginning of a conversation with the remote side and the format is a simple known format such as a GET request from an HTTP web browser. If you want to override this to cause FFST files to be written for any bad data including simple known formats, you can set the **AMQ_BAD_COMMS_DATA_FDCS** environment variable to any value (for example, TRUE) and restart the queue manager.

AMQ_CONVEBCDICNEWLINE



You can use the **AMQ_CONVEBCDICNEWLINE** environment variable to specify how IBM MQ is to convert an EBCDIC NL character into ASCII format. The environment variable takes the same values as the **ConvEBCDICNewline** attribute of the `mqs.ini`, that is, `NL_TO_LF`, `TABLE`, or `ISO` (see All queue managers stanza of the `mqs.ini` file). You can, for example, use the **AMQ_CONVEBCDICNEWLINE** environment variable instead of the **ConvEBCDICNewline** stanza attribute to provide **ConvEBCDICNewline** functionality on the client side in situations where the `mqs.ini` file cannot be used. If both the stanza attribute and the environment variable are set, the stanza attribute takes precedence.

For more information, see [Data conversion between coded character sets](#).

AMQ_DIAGNOSTIC_MSG_SEVERITY

If the **AMQ_DIAGNOSTIC_MSG_SEVERITY** environment variable is set to 1 for an IBM MQ process, this causes the message severity to be appended to the message number as a single uppercase alphabetic character when the IBM MQ process writes a message to an error log or to the console.

The behavior that **AMQ_DIAGNOSTIC_MSG_SEVERITY** enables is set by default. You can turn off this behavior by setting the environment variable to 0.

For more information, see [Using error logs](#).

AMQ_DISABLE_CLIENT_AMS

You can use the **AMQ_DISABLE_CLIENT_AMS** environment variable to disable IBM MQ Advanced Message Security (AMS) at the client if an 2085 (MQRC_UNKNOWN_OBJECT_NAME) error is reported when you are trying to connect to a queue manager from an earlier version of the product and you are using one of the following clients:

- A Java runtime environment (JRE) other than the IBM Java runtime environment (JRE)
- An IBM MQ IBM MQ classes for JMS or IBM MQ classes for Java client.

Note: You cannot use the **AMQ_DISABLE_CLIENT_AMS** environment variable for C clients. You need to use the **MQS_DISABLE_ALL_INTERCEPT** environment variable instead.

For more information, see [Disabling Advanced Message Security at the client](#).

AMQ_DMPMQCFG_QSGDISP_DEFAULT

The inquiries on the disposition of a queue manager that are used by the **dmpmqcfg** command inquire only QSGDISP(QMGR) definitions by default. You can inquire additional definitions by using the **AMQ_DMPMQCFG_QSGDISP_DEFAULT** environment variable, which can be set to one of the following values:

LIVE

Include only, objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

ALL

Include objects defined with QSGDISP(QMGR) and QSGDISP(COPY). If the queue manager is a member of a queue sharing group, QSGDISP(GROUP) and QSGDISP(SHARED) are also included.

COPY

Include only, objects defined with QSGDISP(COPY)

GROUP

Include only, objects defined with QSGDISP(GROUP); the target queue manager must be a member of a queue sharing group.

QMGR

Include only, objects defined with QSGDISP(QMGR). This is the default behavior if you use this environment variable, to match the existing behavior of **dmpmqc.fg**.

PRIVATE

Include only, objects defined with QSGDISP(QMGR) or QSGDISP(COPY).

SHARED

Include only, objects defined with QSGDISP(SHARED).

AMQ_IODELAY, AMQ_IODELAY_INMS and AMQ_IODELAY_FFST



IBM MQ detects when log read and write or input and output operations take longer than anticipated. This might be due to problems with the operating system or storage system and can impact the queue manager performance. From IBM MQ 9.4.0, you can use the **AMQ_IODELAY** environment variables to fine tune diagnostics and timings when the input and output for your queue manager log and storage file system is slow. If you see message AMQ6729W Log I/O operation exceeded threshold in the queue manager error log, investigate the cause and make adjustments accordingly. Use the variables as shown in the following examples:

AMQ_IODELAY

Threshold time in seconds, default is 1 second. If an I/O operation takes longer than this threshold then error message AMQ6729W is reported in IBM MQ log files. The warning message is repeated at most every 10 seconds if the delays continue. You can increase this to suppress errors or decrease to investigate specific performance issues. For example,

```
export AMQ_IODELAY=2000000
```

AMQ_IODELAY_INMS

Change the time measure to microseconds instead of seconds. Use this to set a lower threshold before you get message AMQ6729 in the queue manager log.

```
export AMQ_IODELAY_INMS=YES
```

AMQ_IODELAY_FFST

In addition to the warning message in the error log, a FFST file that contains diagnostic information is generated whenever the threshold is exceeded.

```
export AMQ_IODELAY_FFST=YES
```

Starting the queue manager as in this example, causes an FDC or FFST file to be written if an input/output operation takes longer than 200000 microseconds (0.2s), which is still a relatively generous threshold.

For more information, see [Queue manager health check behavior](#).

AMQ_LDAP_TRACE

If the **AMQ_LDAP_TRACE** environment variable is set to a non-null value, it is possible to switch LDAP client trace on and off without also stopping or starting the queue manager.

For more information, see [Enabling dynamic tracing of LDAP client library code](#).

AMQ_LICENSING_METRIC



Setting the **AMQ_LICENSING_METRIC=VPCMonthlyPeak** environment variable causes the queue manager to upload data relating to monthly VPC license types, instead of the default behavior of uploading data related to hourly container-based licenses.

For more information about configuring IBM MQ for use with the IBM Cloud® Private metering service, see [IBM Cloud Private metering service](#) in the IBM Cloud Private documentation.

AMQ_MQS_INI_LOCATION



On AIX and Linux systems, you can alter the location that is used for the `mqs.ini` file by setting the location of the `mqs.ini` file in the **AMQ_MQS_INI_LOCATION** environment variable. This environment variable must be set at the system level.

For more information about the `mqs.ini` file, including directory locations, see [IBM MQ configuration file, mqs.ini](#).

AMQ_NO_BAD_COMMS_DATA_FDCS

The **AMQ_NO_BAD_COMMS_DATA_FDCS** environment variable is effective when set to any value.

If IBM MQ does not recognize the initial data transmission when attempting to connect a non-IBM MQ client to an IBM MQ TCP/IP listener, this causes the queue manager to write an AMQ9207E error message to the queue manager error logs. A failure data capture (FDC) record is also written. You can suppress the generation of these diagnostic files with the **AMQ_NO_BAD_COMMS_DATA_FDCS** environment variable. When **AMQ_NO_BAD_COMMS_DATA_FDCS** is set to any value (for example, TRUE), this instructs IBM MQ not to generate FFSTs when reporting AMQ9207E error messages on the initial communications flow. To be effective, the environment variable should be set before starting the queue manager and listener processes.

The FDC continues to be generated in the case where a client sends valid IBM MQ protocol flows to the queue manager, and then sends invalid data, as this is indicative of a client issue that warrants further investigation.

Note: The capture of FFSTs when reporting [AMQ9207E](#) error messages on initial communications flows is suppressed by default.

AMQ_NO_IPV6

The **AMQ_NO_IPV6** environment variable is effective when set to any value. When this environment variable is set, it disables the use of IPv6 while attempting a connection.

AMQ_REVERSE_COMMIT_ORDER

The **AMQ_REVERSE_COMMIT_ORDER** environment variable configures a queue manager so that in an XA transaction the IBM MQ queue manager change is committed after the corresponding database update is completed. Applications that read messages from the queues see a message only after the corresponding database update has been completed.

Note: Do not set **AMQ_REVERSE_COMMIT_ORDER** without reading and understanding the scenario that is described in [Isolation level](#).

AMQ_RDQM_NO_PING_CHECKS



When the **AMQ_RDQM_NO_PING_CHECKS** environment variable is set, ICMP pings are disabled in an RDQM configuration. RDQM uses ICMP ping packets to verify remote connections while configuring pacemaker and creating queue managers. By default, this environment variable is not honored for a non-root user with sudo privileges. Either set **AMQ_RDQM_NO_PING_CHECKS** and run the install and configuration commands as root, or update sudoers rules to include the environment variable.

AMQ_SSL_ALLOW_DEFAULT_CERT

When the **AMQ_SSL_ALLOW_DEFAULT_CERT** environment variable is not set, an application can connect to a queue manager with a personal certificate in the client keystore only when the certificate includes the label name of `ibmwebsphermuserid`. When the **AMQ_SSL_ALLOW_DEFAULT_CERT** environment variable is set, the certificate does not require the label name of `ibmwebsphermuserid`. That is, the

certificate that is used to connect to a queue manager can be a default certificate, provided that a default certificate is present in the key repository, and the key repository does not contain a personal certificate with the prefix `ibmwebsphermquserid`.

A value of 1 enables the use of a default certificate.

Instead of using the **AMQ_SSL_ALLOW_DEFAULT_CERT** environment variable, an application can use the **CertificateLabel** setting of the SSL stanza in the `mqclient.ini` file. For more information, see [Digital certificate labels, understanding the requirements](#) and [SSL stanza of the client configuration file](#).

AMQ_SSL_LDAP_SERVER_VERSION

The **AMQ_SSL_LDAP_SERVER_VERSION** environment variable can be used to ensure that either LDAP v2 or LDAP v3 is used by IBM MQ cryptographic components in cases where CRL servers require that a specific version of the LDAP protocol be used.

Set the environment variable to the appropriate value in the environment that is used to start the queue manager or channel:

- To request that LDAP v2 is used, set `AMQ_SSL_LDAP_SERVER_VERSION=2`.
- To request that LDAP v3 is used, set `AMQ_SSL_LDAP_SERVER_VERSION=3`.

This environment variable does not affect LDAP connections established by the IBM MQ queue manager for user authentication or user authorization.

AMQ_TLS_ENVIRONMENT_SCOPE

You can use the **AMQ_TLS_ENVIRONMENT_SCOPE** environment variable to control whether IBM MQ uses a single IBM Global Security Kit (GSKit) environment for the entire process or a IBM Global Security Kit (GSKit) environment per connection. The environment variable takes the same values as the **EnvironmentScope** attribute of the `mqclient.ini`, that is, `PROCESS` or `CONNECTION` (see [SSL stanza of the client configuration file](#)). Setting this environment variable impacts when TLS keystore changes are picked up by client applications. A setting of `CONNECTION` means that TLS keystore changes are picked up on each connection initiated by the application at a cost of increased memory and CPU usage.

AMQ_USE_ZLIBNX



On AIX, the **AMQ_USE_ZLIBNX** environment variable can be used to enable message channel agents (MCA) to use the hardware-accelerated zlibNX library for compression and decompression of message data when using ZLIBFAST or ZLIBHIGH techniques.

Tip: Highly compressible messages that are over 2 KB in size are most likely to benefit from opting to use the zlibNX library, by reducing CPU usage.

The zlibNX library is available in IBM AIX 7.2 with Technology Level 4 Expansion Pack, and later. If the environment variable is set and the zlibNX library (`/usr/opt/zlibNX/lib/libz.a`) is not installed, the message channel agents will use the standard zlib library provided in the IBM MQ for AIX installation.

HOME



On AIX, Linux and IBM i, the **HOME** environment variable specifies the name of the directory that is searched for the `mqclient.ini` file. This file contains configuration information that is used by IBM MQ MQI clients.

For more information, see [IBM MQ MQI client configuration file, mqclient.ini](#) and [Location of the client configuration file](#).

HOMEDRIVE and HOMEPATH

Windows

To be used, both the **HOMEDRIVE** and **HOMEPATH** environment variables must be set. They are used on Windows systems to specify the name of the directory that is searched for the `mqclient.ini` file. This file contains configuration information that is used by IBM MQ MQI clients.

For more information, see [IBM MQ MQI client configuration file, mqclient.ini](#) and [Location of the client configuration file](#).

LDAP_BASEDN

LDAP_BASEDN is the required environment variable for running an LDAP sample program. It specifies the base Distinguished Name for the directory search.

LDAP_HOST

LDAP_HOST is an optional environment variable for running an LDAP sample program. It specifies the name of the host where the LDAP server is running; it defaults to the local host if it is not specified.

LDAP_VERSION

LDAP_VERSION is an optional environment variable for running an LDAP sample program. It specifies the version of the LDAP protocol to be used, and can be either 2 or 3. Most LDAP servers now support version 3 of the protocol; they all support the older version 2. This sample works equally well with either version of the protocol, and if it is not specified it defaults to version 2.

MQCCDTCERTVPOL

V1800

MQCCDTCERTVPOL is an optional environment variable for the CCDT certificate validation policy that can be configured for the IBM MQ classes for JMS client library. You can use **MQCCDTCERTVPOL** to apply the level of validation required for CCDT retrieval over HTTPS connection by specifying one of the following values:

ANY

Verifies that the peer's certificate is valid, signed by a trusted authority, and ignores hostname verification. If a customized `SSLSocketFactory` is provided, then it overrides the default mechanism of Java Secure Socket Extension (JSSE).

NONE

All server certificates are accepted without validation. An internal handler implementation that provides an `SSLSocketFactory` instance trusting all certificates will be used. If a customized `SSLSocketFactory` is provided, it overrides the handler implementation.

HOSTNAMECN

Verifies that the peer's certificate is valid, signed by a trusted authority, and that the certificate distinguished name (including the CN and SAN extensions) matches that of the HTTP server being contacted. If a customized `SSLSocketFactory` is provided, then it overrides the default mechanism of JSSE.

For more information, see [Configuring certificate validation policies in IBM MQ](#).

MQ_CHANNEL_SUPPRESS_INTERVAL

The **MQ_CHANNEL_SUPPRESS_INTERVAL** environment variable specifies the time interval, in seconds, during which the messages defined with **MQ_CHANNEL_SUPPRESS_MSGS** are to be suppressed from being written to the error log, together with the number of times that a message will be allowed to occur during the specified time interval before being suppressed. The default value is 60,5 which means that any further occurrences of a given message are suppressed after the first five occurrences of that message in

a 60 second interval. For more information, see [Suppressing channel error messages from error logs on Multiplatforms](#).

The **MQ_CHANNEL_SUPPRESS_INTERVAL** environment variable is comparable to [SuppressInterval](#) in the `qm.ini` file.

MQ_CHANNEL_SUPPRESS_MSGS

The **MQ_CHANNEL_SUPPRESS_MSGS** environment variable suppresses channel error messages in the error log. You can specify a list of messages that are suppressed. **MQ_CHANNEL_SUPPRESS_MSGS** is used in conjunction with **MQ_CHANNEL_SUPPRESS_INTERVAL**, which specifies the number of times that each message appears before being suppressed and the length of time that messages are suppressed for. For more information, see [Suppressing channel error messages from error logs on Multiplatforms](#).

The **MQ_CHANNEL_SUPPRESS_MSGS** environment variable is comparable to [SuppressMessage](#) in the `qm.ini` file, except that you can suppress any channel message by using the environment variable, whereas there is a restrictive list for the `qm.ini` method.

MQ_CONNECT_TYPE



On Multiplatforms, you can use the **MQ_CONNECT_TYPE** environment variable in combination with the type of binding specified in the Options field of the MQCNO structure that is used on an MQCONN call. **MQ_CONNECT_TYPE** only has any effect for STANDARD bindings. For other bindings, **MQ_CONNECT_TYPE** is ignored.

For more information, see [Use of MQCONN call options with MQ_CONNECT_TYPE](#).

MQ_CROSS_QUEUE_ORDER_ALL

When you set the **MQ_CROSS_QUEUE_ORDER_ALL** environment variable to a non-zero value, the message put order is maintained in a unit of work. This means that, if messages in a Unit of Work (UoW) are put onto multiple queues (for example, Q1, then Q2), when an MQCMIT is issued, the messages are delivered and made available in the same queue order in which they were PUT.

In a multi-queue manager environment, **MQ_CROSS_QUEUE_ORDER_ALL** must exist and have a non-empty value on both the sending and receiving side before each queue manager is started.

MQ_EPHEMERAL_PREFIX

The **MQ_EPHEMERAL_PREFIX** environment variable specifies the path to the queue manager's ephemeral directory, within which ephemeral queue manager data is kept, while the queue manager is running.

As an alternative to changing the ephemeral prefix by changing the **EphemeralPrefix** attribute in the **DefaultEphemeralPrefix** attribute of the AllQueueManagers stanza of the `mqs.ini` file, you can use the environment variable **MQ_EPHEMERAL_PREFIX** to override the **EphemeralPrefix** for the `crtmqm` command. For more information, see [Configurable ephemeral directory](#).

MQ_FILE_PATH



The **MQ_FILE_PATH** environment variable is configured during the installation of the runtime package on the Windows platform. This environment variable contains the same data as the following key in the Windows registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\Installation\InstallationName\FilePath
```

For more information, see [setmqenv \(set IBM MQ environment\)](#) and [crtmqenv \(create IBM MQ environment\)](#).

MQ_JAVA_DATA_PATH

The **MQ_JAVA_DATA_PATH** environment variable specifies the directory for log and trace output for the IBM MQ classes for JMS and IBM MQ classes for Jakarta Messaging and IBM MQ classes for Java. It is used by the scripts provided with IBM MQ classes for JMS and IBM MQ classes for Jakarta Messaging and IBM MQ classes for Java.

For more information, see [Setting environment variables for IBM MQ classes for JMS/Jakarta Messaging](#) and [Environment variables relevant to IBM MQ classes for Java](#).

MQ_JAVA_INSTALL_PATH

The **MQ_JAVA_INSTALL_PATH** environment variable specifies the directory where the IBM MQ classes for JMS and IBM MQ classes for Jakarta Messaging as shown in [What is installed for IBM MQ classes for JMS](#), and the IBM MQ classes for Java as shown in [IBM MQ classes for Java installation directories](#) are installed.

For more information, see [Setting environment variables for IBM MQ classes for JMS/Jakarta Messaging](#) and [Environment variables relevant to IBM MQ classes for Java](#).

MQ_JAVA_LIB_PATH

The **MQ_JAVA_LIB_PATH** environment variable specifies the directory where the IBM MQ classes for JMS and IBM MQ classes for Jakarta Messaging, and the IBM MQ classes for Java libraries are stored. Some scripts, for example, IVTRun, that are supplied with the IBM MQ classes for JMS and IBM MQ classes for Jakarta Messaging or the IBM MQ classes for Java use this environment variable.

For more information, see [Setting environment variables for IBM MQ classes for JMS/Jakarta Messaging](#) and [Environment variables relevant to IBM MQ classes for Java](#).

MQ_LCLADDR



On [Multiplatforms](#), you can set the **MQ_LCLADDR** environment variable to define a default local address value that is 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 must match that of MQSC attribute **LOCLADDR**. For more information, see [LOCLADDR \(Local Address\)](#) and [“Working with auto-defined cluster-sender channels”](#) on page 312.

MQ_OVERRIDE_DATA_PATH

You can use the **MQ_OVERRIDE_DATA_PATH** environment variable to change the default directory of the IBM MQ data path.

MQ_SET_NODELAYACK



The **MQ_SET_NODELAYACK** environment variable switches off TCP delayed acknowledgment on AIX.

When you set this environment variable, the setting switches off TCP delayed acknowledgment by calling the operating system's setsockopt call with the TCP_NODELAYACK option. Only AIX supports this function, so the **MQ_SET_NODELAYACK** environment variable only has an effect on AIX.

MQ_USER_NAME



If the user has no password file entry, or the user has a name that matches its user identifier, you can use the **MQ_USER_NAME** environment variable to allow an IBM MQ non-install image on Linux to choose the user name. This is needed, for example, to use publish/subscribe hierarchies in OpenShift.

The value of **MQ_USER_NAME** must not match a pre-existing user on the system, and must be less than or equal to 12 bytes.

MQAPI_TRACE_LOGFILE

The sample API exit program generates an MQI trace to a user-specified file with a prefix that is defined in the **MQAPI_TRACE_LOGFILE** environment variable.

For more information, see [The API exit sample program](#).

MQAPPLNAME



If the application name has not yet been chosen, you can use the **MQAPPLNAME** environment variable as the name to be used to identify the connection to a queue manager. Only the first 28 characters are used, and they must not be all blanks or nulls.

For more information, see [Using the application name in supported programming languages](#).

MQCCSID

The **MQCCSID** environment variable specifies the coded character set number to be used and overrides the CCSID value with which the server has been configured. **MQCCSID** can be used to override the native CCSID of an application and specify the coded character set number to be used, for example if the native CCSID is an unsupported CCSID or is not the required CCSID.

To set **MQCCSID**, use one of the following commands:

-   On AIX and Linux:

```
export MQCCSID=number
```

-  On Windows:

```
SET MQCCSID=number
```

-  On IBM i:

```
ADDENVVAR ENVVAR(MQCCSID) VALUE(number)
```

For more information, see [Choosing client or server CCSID](#).

MQCCDTURL

The **MQCCDTURL** environment variable provides the equivalent capability to setting a combination of the **MQCHLLIB** and **MQCHLTAB** environment variables. Use this variable to provide a file, ftp, or http URL as a single value from which a client channel definition table can be obtained for native programs (C, COBOL, or C++ applications) that are connecting as clients.  From IBM MQ 10.0.0, https URLs are also supported.

Notes:

- Using environment variables to provide the URL has no effect for Java, JMS or managed .NET applications.
- The maximum length of the URL is the maximum file path length of the operating system.

IBM MQ supports retrieving a CCDT from a file, ftp, or http URL,  or, from IBM MQ 10.0.0, an https URL. However, **MQCCDTURL** does not accept an FTP value. It only accepts a URL value, or a file specified using the 'file:/' protocol.

For example, when using **MQCCDTURL** in place of **MQCHLLIB** and **MQCHLTAB** with a local file on AIX and Linux, this command:

```
export MQCCDTURL=file:///var/mqm/qmgrs/QMGR/@ipcc/MYCHL.TAB
```

is equivalent to:

```
export MQCHLLIB=/var/mqm/qmgrs/QMGR/@ipcc
export MQCHLTAB=MYCHL.TAB
```

You can also specify a JSON file as shown in this example for Windows:

```
set MQCCDTURL=file:/c:/mq-channels/CCDT-QMGR1.json
```

is equivalent to:

```
set MQCHLLIB=C:\mq-channels
set MQCHLTAB=CCDT-QMGR1.json
```

For more information, see [URL access to the CCDT](#).

MQCERTLABL

The **MQCERTLABL** environment variable defines the certificate label of a channel definition for IBM MQ to use to locate a personal certificate that is sent during a TLS handshake.

For more information, see [Digital certificate labels, understanding the requirements](#).

MQCERTVPOL

The **MQCERTVPOL** environment variable specifies the type of certificate validation policy to be used. This environment variable overrides the **CertificateValPolicy** attribute in the SSL stanza of the client configuration file.

MQCERTVPOL can be set to one of the following values:

ANY

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

RFC5280

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

NONE

Do not use any certificate validation policy. This setting is for client applications only.

To set **MQCERTVPOL**, use one of these commands:

-   For AIX and Linux systems:

```
export MQCERTVPOL= value
```

-  For Windows systems:

```
SET MQCERTVPOL= value
```

-  For IBM i systems:

```
ADDENVVAR ENVVAR(MQCERTVPOL) VALUE(value)
```

For more information, see [Certificate validation policies in IBM MQ](#) and [Configuring certificate validation policies in IBM MQ](#).

MQCHLLIB

The **MQCHLLIB** environment variable specifies the directory path to the file that contains 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.

To set **MQCHLLIB**, use one of these commands:

- **Windows** On Windows:

```
SET MQCHLLIB=pathname
```

For example:

```
SET MQCHLLIB=C:\wmqtest
```

- **Linux** **AIX** For AIX and Linux systems:

```
export MQCHLLIB=pathname
```

- **IBM i** For IBM i:

```
ADDENVVAR ENVVAR(MQCHLLIB) VALUE(pathname)
```

If **MQCHLLIB** is not set, the path for the client defaults to:

- **Linux** **AIX** On AIX and Linux: `/var/mqm/`
- **Windows** On Windows: `MQ_INSTALLATION_PATH`
- **IBM i** On 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.

- **Linux** **AIX** On AIX and Linux: `datapath/@ipcc`
- **Windows** On Windows: `datapath\@ipcc`
- **IBM i** On IBM i: `datapath/&ipcc`

Or:

- **Linux** **AIX** On AIX and Linux: `/prefix/qmgrs/qmgrname/@ipcc`
- **Windows** On Windows: `MQ_INSTALLATION_PATH\data\qmgrs\qmgrname\@ipcc`
- **IBM i** On 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 one of the following values:
 - **Linux** **AIX** `/var/mqm` on AIX and Linux systems.
 - **IBM i** `/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.

- Where the queue manager stanza is defined depends on the platform:

-    In the `mqsc.ini` file on IBM i, AIX and Linux.
-  In the registry on Windows.

Notes:

1.  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. **MQCHLLIB** can contain a URL which works in combination with the **MQCHLTAB** environment variable (see [“URL access to the CCDT”](#) on page 53).
4. Environment variables, such as **MQCHLLIB**, can be scoped to a process, or a job, or system-wide, in a platform-specific way.
5. 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.
6. 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.
7. 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.
8. The command, **rcrmqobj -t clchltab** deletes and re-creates the CCDT file. The file is re-created with only the client-connection definitions created on the queue manager that the command is running against.
9. 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.
10. The path for **MQCHLLIB** does not need quotation marks.

For more information, see [Locations for the CCDT](#), [URL access to the CCDT](#), and [Connecting client applications to queue managers using environment variables](#).

MQCHLTAB

The **MQCHLTAB** environment variable specifies the name of the file that contains the client channel definition table (CCDT). The default file name is **AMQCLCHL.TAB**.

To set **MQCHLTAB**, use one of these commands:

-   On AIX and Linux:

```
export MQCHLTAB=filename
```

-  On Windows:

```
SET MQCHLTAB=filename
```

-  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.

For more information, see [Locations for the CCDT](#), [URL access to the CCDT](#), and [Connecting client applications to queue managers using environment variables](#).

MQCLNTCF

The **MQCLNTCF** environment variable specifies the location of the IBM MQ MQI client configuration file. This file contains configuration information that is used by IBM MQ MQI clients.

Examples of valid formats are listed below. You can choose a filename other than `mqclient.ini`. You can choose any valid path on your system that is readable by the client at run-time.

  These are examples of valid settings for MQCLNTCF on AIX and Linux systems when your file is located at `/data/mqclient.ini`.

```
export MQCLNTCF=/data/mqclient.ini
export MQCLNTCF=file:/data/mqclient.ini
export MQCLNTCF=file:///data/mqclient.ini
export MQCLNTCF=file://localhost//data/mqclient.ini
```

 These are examples of valid settings for MQCLNTCF on Windows systems when your file is located at `C:\data\mqclient.ini`.

```
set MQCLNTCF=C:\data\mqclient.ini
set MQCLNTCF=file:C:\data\mqclient.ini
set MQCLNTCF=file:///C:\data\mqclient.ini
set MQCLNTCF=file://localhost/C:\data\mqclient.ini
```

 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.

For more information, see [IBM MQ MQI client configuration file, mqclient.ini](#) and [Location of the client configuration file](#).

MQDOTNET_TRACE_ON

The **MQDOTNET_TRACE_ON** environment variable is used to enable trace for IBM MQ .NET redistributable clients. Values equal to and less than 0 do not enable trace, 1 enables default trace and values greater than 1 enable detail tracing.

For more information, see [Tracing IBM MQ .NET applications](#) and [Tracing IBM MQ .NET applications using environment variables](#).

MQIPADDRV

The **MQIPADDRV** environment variable 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** and the **IPAddressVersion** attribute of the TCP stanza of the client configuration file. If the environment variable is not set, "MQIPADDR_IPV4" is assumed.

To set **MQIPADDRV**, use one of these commands:

-   On AIX and Linux:

```
export MQIPADDRV=MQIPADDR_IPV4|MQIPADDR_IPV6"/>
```

-  On Windows:

```
SET MQIPADDRV=MQIPADDR_IPV4|MQIPADDR_IPV6
```

-  On IBM i:

```
ADDENVVAR ENVVAR(MQIPADDRV) VALUE(MQIPADDR_IPV4|MQIPADDR_IPV6)
```

MQKEYRPWD

When you set the **MQKEYRPWD** environment variable, it specifies the password for the key repository that holds the digital certificate belonging to the user. If **MQKEYRPWD** is used, you must encrypt the password before you set the value of the environment variable.

To set **MQKEYRPWD**, use one of these commands:

-   On AIX and Linux systems:

```
export MQKEYRPWD=passphrase
```

-  On Windows systems:

```
SET MQKEYRPWD=passphrase
```

-  On IBM i:

```
ADDENVVAR ENVVAR(MQKEYRPWD) VALUE(passphrase)
```

There is no default value for this environment variable.

For more information, see

-  [Supplying the key repository password for an IBM MQ MQI client on AIX, Linux, and Windows and Encrypting the key repository password](#)
-  [Supplying the key repository password for an IBM MQ MQI client on IBM i and Encrypting the key repository password.](#)

MLICENSE

Linux

On Linux systems, you can use the **MLICENSE** environment variable to accept or view an IBM MQ license after you install the product.

For more information about why you might want or need to accept the license after installation, see [Accepting the license on IBM MQ for Linux](#).

Important: If you use the **MLICENSE** environment variable to accept or view an IBM MQ license after you install the product, bear in mind the following points:

- Accepting the license by setting the **MLICENSE** environment variable is only valid for the current session.
- Running the **dspmqr** command still shows the license as not accepted if you run it after accepting the license by setting the **MLICENSE** environment variable.

The **MLICENSE** environment variable can be set to one of two values:

accept

Accept the license post-installation.

To accept the license post-installation, use this command:

```
export MQLICENSE=accept
```

view

Display the license, if the license has been accepted.

To view the license, use this command:

```
export MQLICENSE=view
```

Note: You can also use the following commands to accept and display the license:

- [mqlicense \(accept license post installation\)](#)
- [dspmqlic \(display IBM MQ license\)](#)

MQMAXERRORLOGSIZE



The **MQMAXERRORLOGSIZE** environment variable specifies the size of the queue manager error log that is copied to the backup.

For more information, see [Using error logs](#).

MQNAME



The **MQNAME** environment variable specifies the local NetBIOS name that the IBM MQ processes can use. A NetBIOS connection applies only to a client and server running Windows.

To set **MQNAME**, use this command:

```
SET MQNAME=Your_env_Name
```

For example:

```
SET MQNAME=CLIENT1
```

Some NetBIOS implementations require a unique name, set by **MQNAME**, for each application if you are running multiple IBM MQ applications simultaneously on the IBM MQ MQI client.

For more information, see [Defining the IBM MQ local NetBIOS name](#).

MQNOREMPOOL

When you set the **MQNOREMPOOL** environment variable, it switches off channel pooling and causes channels to run as threads of the listener.

For more information, see [MCATYPE \(Message channel agent type\)](#).

MQPSE_TRACE_LOGFILE

You use the **MQPSE_TRACE_LOGFILE** environment variable when you run the Publish Exit Sample Program AMQSPSE0, which is a sample C program of an exit to intercept a publication before it is delivered to a subscriber. In the application process to be traced, this environment variable describes where the trace files must be written to.

For more information, see [The Publish Exit sample program](#).

MQS_AMSCRED_KEYFILE

You can use the **MQS_AMSCRED_KEYFILE** environment variable to override or provide the initial key file to use at run time of IBM MQ Advanced Message Security (AMS) applications, or when you are protecting a keystore configuration file by using the [runamscred](#) command.

For more information, see [Using keystores and certificates with AMS](#) and [Protecting passwords in IBM MQ component configuration files](#).

MQS_DISABLE_ALL_INTERCEPT

You can use the **MQS_DISABLE_ALL_INTERCEPT** environment variable to disable IBM MQ Advanced Message Security (AMS) if an 2085 (MQRC_UNKNOWN_OBJECT_NAME) error is reported when you are trying to connect to a queue manager from an earlier version of the product and you are using IBM MQ with native C clients.

Note: You can use the **MQS_DISABLE_ALL_INTERCEPT** environment variable for C clients only. For Java clients, you need to use the [AMQ_DISABLE_CLIENT_AMS](#) environment variable instead.

For more information, see [Disabling Advanced Message Security at the client](#).

MQS_IPC_HOST

Since IPC file system objects have to be distinguished by system, a subdirectory for each system that the queue manager runs on is added to the directory path. If the generated value of the host name creates a problem, you can set the host name by using the **MQS_IPC_HOST** environment variable.

For more information, see [Sharing IBM MQ files on Multiplatforms](#).

MQS_KEystore_CONF

The **MQS_KEystore_CONF** environment variable specifies the location of the keystore configuration file for IBM MQ Advanced Message Security (AMS), if the file is not in the default location of *home_directory/.mq/keystore.conf*.

For more information, see [Using keystores and certificates with AMS](#).

If you are having problems on Managed File Transfer, see [Troubleshooting when MFT does not read keystore properties for AMS](#).

MQS_MQI_KEYFILE

When you set the **MQS_MQI_KEYFILE** environment variable, it specifies the location of an initial key file which contains the initial key to be used for password protection operations. If the initial key file is not specified, the default initial key is used by the IBM MQ password protection system.

To set **MQS_MQI_KEYFILE**, use one of these commands:

-  On AIX and Linux systems:

```
export MQS_MQI_KEYFILE=key file location
```

-  On Windows systems:

```
SET MQS_MQI_KEYFILE=key file location
```

-  On IBM i:

```
ADDENVVAR ENVVAR(MQS_MQI_KEYFILE) VALUE(key file location)
```

For more information, see [Supplying an initial key for an IBM MQ MQI client on AIX, Linux, and Windows](#) and [Supplying an initial key for an IBM MQ MQI client on IBM i](#).

MQS_SSLCRYP_KEYFILE

The **MQS_SSLCRYP_KEYFILE** environment variable is an alternative way of specifying the full path and name of the file containing the initial key used to encrypt the password in the PKCS #11 cryptographic hardware configuration string, instead of specifying it with the **SSLCryptoHardwareKeyFile** attribute in the **SSL** stanza of the **qm.ini**. **MQS_SSLCRYP_KEYFILE** has a higher priority than the **qm.ini** file so its value takes precedence over any other value. For more information, see [IBM MQ clients using cryptographic hardware](#).

MQS_TRACE_OPTIONS



For selective component tracing on AIX, use the **MQS_TRACE_OPTIONS** environment variable to activate the high detail and parameter tracing functions individually.

Note: Only set the **MQS_TRACE_OPTIONS** environment variable if you have been instructed to do so by IBM Support.

For more information, see [Tracing on AIX and Linux](#).

MQS_GETGROUPLIST_API



The **MQS_GETGROUPLIST_API** environment variable improves performance, by using the faster **getgrouplist** API instead of the slower **getgrent** API. It helps avoid queue manager timeouts if your enterprise has thousands of groups to search.

To enable the **MQS_GETGROUPLIST_API** environment variable, you need to do the following:

1. Stop the queue manager.
2. Issue the following command:

```
export MQS_GETGROUPLIST_API=1
```

3. Restart the queue manager.

This change can also resolve RC2035 errors that are related to user group access. For more information on the **MQS_GETGROUPLIST_API** variable, see [Solving certain group membership problems on Linux](#).

MQSERVER

The **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.

Note: You cannot use **MQSERVER** to define a TLS channel or a channel with channel exits. For more information on how to define a TLS channel, see [Protecting channels with TLS](#).

The following examples show how to set **MQSERVER**:

-   On AIX and Linux:

```
export MQSERVER='SYSTEM.DEF.SVRCONN/TCP/AMACHINE.ACOMPANY.COM(1414)'
```

-  On Windows:

```
SET MQSERVER=SYSTEM.DEF.SVRCONN/TCP/AMACHINE.ACOMPANY.COM(1414)
```

-  On IBM i:

```
ADDENVVAR ENVVAR(MQSERVER) VALUE('SYSTEM.DEF.SVRCONN/TCP/AMACHINE.ACOMPANY.COM(1414)')
```

Note:

- The channel name 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.
- The transport type can be one of LU62 , TCP , NETBIOS, SPX, depending on your IBM MQ client platform.
- The connection name must be a fully-qualified network name., for example AMACHINE.ACOMPANY.COM(1414).
- The connection name 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 connection name 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 connection name list to specify different queue manager instances.

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).

For more information, see [Creating a client-connection channel on the IBM MQ MQI client using MQSERVER](#).

MQSNOAUT



Warning: This functionality is not recommended.

When you set the **MQSNOAUT** environment variable to any value it disables the object authority manager (OAM) and prevents any security checking. This might be suitable for a test environment. This includes both authorization and connection authentication functionality. TLS, Channel Authentication Records and Security Exits are unaffected.

The **MQSNOAUT** environment variable only takes effect when a queue manager is created.



Warning: To enable the OAM, you must delete the queue manager, delete the environment variable, and then recreate the queue manager without specifying **MQSNOAUT**.

For more information, see [Preventing security access checks on AIX, Linux, and Windows systems](#).

MQSPREFIX

As an alternative to changing the default prefix, you can use the **MQSPREFIX** environment variable to override the **DefaultPrefix** for the **crtmqm** command.

For more information, see [IBM MQ file names](#) and [AllQueueManagers stanza of the mqs.ini file](#).

MQSSLCRYP



The **MQSSLCRYP** environment variable holds a parameter string that you can use to configure the cryptographic hardware present on the system.

The permitted values are the same as for the [SSLCryptoHardware](#) field in the SSL stanza of the client configuration file.

To set **MQSSLCRYP**, use one of these commands:

-   On AIX and Linux systems:

```
export MQSSLCRYP=string
```

- **Windows** On Windows systems:

```
SET MQSSLCryp=string
```

For more information, see [Configuring for cryptographic hardware on AIX, Linux, and Windows](#) and [IBM MQ clients that use cryptographic hardware in Protecting passwords in IBM MQ component configuration files](#).

MQSSLFIPS

V10.00 The **MQSSLFIPS** environment variable specifies whether only FIPS certified cryptographic libraries and algorithms are to be used if cryptography is carried out in IBM MQ.

You can set this environment variable to YES or NO. The default is NO. These values are the same as for the **SSLFIPS** parameter of the **ALTER QMGR** command.

To set **MQSSLFIPS**, use one of these commands:

- **Linux** **AIX** On AIX and Linux systems:

```
export MQSSLFIPS=YES|NO
```

- **Windows** On Windows systems:

```
SET MQSSLFIPS=YES|NO
```

- **IBM i** On IBM i:

```
ADDENVVAR ENVVAR(MQSSLFIPS) VALUE(YES|NO)
```

The use of FIPS certified algorithms is affected by the use of cryptographic hardware. For more information, see [Specifying that only FIPS certified CipherSpecs are used at run time on the MQI client](#).

MQSSLKEYR

The **MQSSLKEYR** environment variable specifies the location of the key repository that holds the digital certificate belonging to the user.

Specify the full path and the file name of the key repository. If the file suffix is not specified, it is assumed to be .kdb.

To set **MQSSLKEYR**, use one of these commands:

- **Linux** **AIX** On AIX and Linux systems:

```
export MQSSLKEYR=pathname
```

- **Windows** On Windows systems:

```
SET MQSSLKEYR=pathname
```

- **IBM i** On IBM i:

```
ADDENVVAR ENVVAR(MQSSLKEYR) VALUE(pathname)
```

There is no default value for this environment variable.

For more information, see the **SSLKEYR** parameter of the **ALTER QMGR** command.

MQSSLPROXY

The **MQSSLPROXY** environment variable specifies the host name and port number of the HTTP proxy server to be used by GSKit for OCSP checks.

To set **MQSSLPROXY**, use one of these commands:

- ▶ **Linux** ▶ **AIX** On AIX and Linux systems:

```
export MQSSLPROXY="string"
```

- ▶ **Windows** On Windows systems:

```
SET MQSSLPROXY= string
```

The string that you specify with **MQSSLPROXY** can be 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.

▶ **Linux** ▶ **AIX** For example, on AIX and Linux systems, you can use the one of the following commands:

- `export MQSSLPROXY="proxy.example.com(80)"`
- `export MQSSLPROXY="127.0.0.1"`

For more information, see [Working with Online Certificate Status Protocol \(OCSP\)](#).

MQSSLRESET

The **MQSSLRESET** environment variable specifies the number of unencrypted bytes sent and received on a TLS channel before the TLS secret key is renegotiated. 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 a TLS secret key reset count in the range 1 byte through 32 KB, 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 TLS secret key reset values.

To set **MQSSLRESET**, use one of these commands:

- ▶ **Linux** ▶ **AIX** On AIX and Linux systems:

```
export MQSSLRESET=integer
```

- ▶ **Windows** On Windows systems:

```
SET MQSSLRESET=integer
```

- ▶ **IBM i** On IBM i:

```
ADDENVVAR ENVVAR(MQSSLRESET) VALUE(integer)
```

For more information, see [Resetting SSL and TLS secret keys](#).

MQSUITEB

▶ **ALW**

You can configure IBM MQ to operate in compliance with the NSA Suite B standard on AIX, Linux, and Windows platforms.

The **MQSUIEB** environment variable specifies whether Suite B compliant cryptography is to be used. If Suite B cryptography is to be used, you can specify the strength of the cryptography by setting **MQSUIEB** to one of the following:

- NONE
- 128_BIT, 192_BIT
- 128_BIT
- 192_BIT

You can specify multiple values using a comma separated list. Using the value NONE with any other value is invalid.

For more information, see [Configuring IBM MQ for Suite B](#).

MQTCPTIMEOUT

The **MQTCPTIMEOUT** environment variable specifies how long IBM MQ waits for a TCP connect call.

MQTRACEPATH



When taking long traces of an IBM MQ queue manager or a C-based IBM MQ client application, you can mitigate any potential disk space issues by using the environment variable **MQTRACEPATH** to point to a directory that has more disk space.

- On AIX and Linux:

```
set MQTRACEPATH=/other/mqtrace
```

- On Windows:

```
set MQTRACEPATH=D:\mqtrace
```

For more information, see [Tracing on AIX and Linux](#) and [Tracing on Windows](#).

If you are tracing IBM MQ classes for .NET Framework clients using an application configuration file, you can **MQTRACEPATH** to point to a folder where the trace files will be created. For more information, see [Tracing IBM MQ classes for .NET Framework clients using an application configuration file](#).

ODQ_MSG

If you use a dead-letter queue handler that is different from **runmqdlq**, the source of the sample, **amqsd1q**, is available for you to use as your base. The sample is like the dead-letter handler provided within the product but trace and error reporting are different. Use the **ODQ_MSG** environment variable to set the name of the file containing error and information messages. The file that is provided is called **amqsd1q.msg**.

For more information, see [Dead-letter queue handler sample](#).

ODQ_TRACE

If you use a dead-letter queue handler that is different from **runmqdlq**, the source of the sample, **amqsd1q**, is available for you to use as your base. The sample is like the dead-letter handler provided within the product but trace and error reporting are different. To enable tracing, set the **ODQ_TRACE** environment variable to YES or yes.

For more information, see [Dead-letter queue handler sample](#).

TOKENCERTVPOL



TOKENCERTVPOL is an optional environment variable for JWT certificate validation policy that can be configured for the IBM MQ classes for JMS client library. You can use **TOKENCERTVPOL** to apply the level of validation required for the token retrieval over HTTPS connection. You can use it to apply the level of validation required for CCDT retrieval over HTTPS connection by specifying one of the following values:

ANY

Verifies that the peer's certificate is valid, signed by a trusted authority, and ignores hostname verification. If a customized SSLSocketFactory is provided, then it overrides the default mechanism of Java Secure Socket Extension (JSSE).

NONE

All server certificates are accepted without validation. An internal handler implementation that provides an SSLSocketFactory instance trusting all certificates will be used. If a customized SSLSocketFactory is provided, it overrides the handler implementation.

HOSTNAMECN

Verifies that the peer's certificate is valid, signed by a trusted authority, and that the certificate distinguished name (including the CN and SAN extensions) matches that of the HTTP server being contacted. If a customized SSLSocketFactory is provided, then it overrides the default mechanism of JSSE.

For more information, see [Configuring certificate validation policies in IBM MQ](#).

WCF_TRACE_ON

Two different trace methods are available for the WCF custom channel. These two trace methods are activated either independently or together. Each method produces its own trace file, so when both trace methods have been activated, two trace output files are generated. There are four combinations for enabling and disabling the two different trace methods. As well as these combinations to enable WCF trace, the XMS .NET trace can be enabled by using the **WCF_TRACE_ON** environment variable.

For more information, see [Tracing the WCF custom channel for IBM MQ](#).

WMQSOAP_HOME

The **WMQSOAP_HOME** environment variable is used when completing additional configuration steps after the .NET SOAP over JMS service hosting environment is correctly installed and configured in IBM MQ. It is accessible from a local queue manager.

For more information, see [WCF client to a .NET service hosted by IBM MQ sample](#) and [WCF client to an Axis Java service hosted by IBM MQ sample](#).

XMS_TRACE_ON, XMS_TRACE_FILE_NUMBER, XMS_TRACE_FILE_PATH, XMS_TRACE_FILE_SIZE, XMS_TRACE_FORMAT, and XMS_TRACE_SPECIFICATION

If you are using IBM MQ classes for XMS .NET Framework, you can configure trace from an application configuration file as well as from the XMS environment variables. If you are using IBM MQ classes for XMS .NET (.NET Standard and .NET 6 libraries), you must configure trace from the XMS environment variables. Trace is normally used under the guidance of IBM Support.

To enable and configure trace for an XMS .NET application, set the following environment variables before running the application:

XMS_TRACE_ON

If the **XMS_TRACE_ON** environment variable is set, all trace is enabled by default.

XMS_TRACE_FILE_NUMBER

The **XMS_TRACE_FILE_NUMBER** environment variable specifies the number of trace files that are to be retained. The default is 4 (one active file and 3 archive files). The minimum number allowed is 2.

XMS_TRACE_FILE_PATH

The **XMS_TRACE_FILE_PATH** environment variable specifies the fully qualified path name of the directory that trace and FFDC records are written to, if you want these records to be written to an alternative location from the current working directory.

XMS_TRACE_FILE_SIZE

The **XMS_TRACE_FILE_SIZE** environment variable specifies the maximum allowed size of the trace file. When a file reaches this size, it is archived and renamed. The default maximum is 20 MB. File size is specified in bytes. For example, **XMS_TRACE_FILE_SIZE**="20000000".

XMS_TRACE_FORMAT

The **XMS_TRACE_FORMAT** environment variable specifies the required trace format, which can be either BASIC or ADVANCED.

XMS_TRACE_SPECIFICATION

The **XMS_TRACE_SPECIFICATION** environment variable overrides the trace settings defined in the Trace section of an application configuration file. **XMS_TRACE_SPECIFICATION** applies to IBM MQ classes for XMS .NET Framework only.

For more information, see [Tracing XMS .NET applications](#) and [Tracing XMS .NET applications using XMS environment variables](#).

Changing IBM MQ configuration information in .ini files on Multiplatforms

You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation by editing the information in the configuration (.ini) files. You can also change configuration options for IBM MQ MQI clients.

About this task

You can change IBM MQ configuration information at the level of the node or the queue manager by changing the values that are specified on a set of configuration attributes (or parameters) that govern IBM MQ.

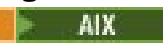
A configuration file (or stanza file) contains one or more stanzas, which are groups of lines in the .ini file that together have a common function or define part of a system, such as log functions, channel functions, and installable services. You can modify IBM MQ configuration attributes within the following configuration files:

IBM MQ configuration file, mqs.ini

The mqs.ini file effects changes on the node as a whole.

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.

Installation configuration file, mqinst.ini

  On AIX and Linux systems, the installation configuration file, mqinst.ini, contains information about all the IBM MQ installations. The mqinst.ini file must not be edited or referenced directly since its format is not fixed, and could change. Instead, you must edit it using commands.

Queue manager configuration file, qm.ini

The qm.ini file effects changes for specific queue managers. There is one qm.ini file for each queue manager on the node.

IBM MQ MQI client configuration file, mqclient.ini

Configuration options for IBM MQ MQI clients are held separately, in the client configuration file, which is generally named mqclient.ini.

Activity trace configuration file, mqat.ini

The mqat.ini file is used to configure activity trace behavior.

You might need to edit a configuration file if, for example:

- You lose a configuration file. (Recover it 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 IBM Support.

Important: Any changes that you make to a configuration file usually do not take effect until the next time the queue manager is started.

Points to note about editing configuration files:

- The values of the attributes of a configuration file are set according to the following priorities:
 - Parameters that are entered on the command line take precedence over values that are defined in the configuration files.
 - Values that are defined in the `qm.ini` files take precedence over values that are defined in the `mqs.ini` file.
- After installation, you can edit the default values in the IBM MQ configuration files.
- When backing up a queue manager, remember to include both its configuration file (`qm.ini`) and the central IBM MQ configuration file (`mqs.ini`).
- If you set an incorrect value on a configuration file attribute, the effect is the same as missing out the attribute entirely. The value is ignored and an operator message is issued to indicate the problem.
-  On IBM i, the `.ini` files are stream files resident in the IFS.
- There are a number of syntax rules for the format of the `mqat.ini` file. For more information, see Application activity trace [Configuring activity trace behavior using mqat.ini](#).

Procedure

1. Before you edit a configuration file, back it up so that you have a copy that you can revert to, if the need arises.

Note:

The backup should not be called `<existing file>.ini.bak` in the same directory as this is reserved for internal use.

2. Edit the `.ini` configuration file in one of the following ways:

- Manually by using a standard text editor. 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. The character is then used as part of the configuration data.
- Automatically, by using commands that change the configuration of queue managers on the node. For more information, see [Commands reference](#).

 For example, the Windows specific command **amqmdain** will update a subset of the `qm.ini` properties automatically. For more information, see [amqmdain](#).

-   On Linux (x86 and x86-64) and Windows, you can update a subset of the `qm.ini` properties by using IBM MQ Explorer. For more information, see [Configuring IBM MQ using MQ Explorer](#).

Note: Because there are significant implications to changing installable services and their components, the installable services are read-only in the IBM MQ Explorer. Therefore, you must make any changes to installable services by editing the `qm.ini` file. For more information, see [“Service stanza of the qm.ini file” on page 153](#).

Related tasks

[Administering IBM MQ](#)

Multi 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.

Note: For more information on how and when to edit the `mqs.ini` file and on when any changes that you make to the file take effect, see [“Changing IBM MQ configuration information in .ini files on Multiplatforms”](#) on page 94.

Directory locations

Linux **AIX** On AIX and Linux, the data directory and log directory are always `/var/mqm` and `/var/mqm/log` respectively.

Windows 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. The installation configuration information, which is contained in `mqinst.ini` on AIX and Linux systems, is also in the registry, as there is no `mqinst.ini` file on Windows (see [“Installation configuration file, mqinst.ini”](#) on page 171).

Windows The `mqs.ini` file for Windows systems is given by the `WorkPath` specified in the `HKLM\SOFTWARE\IBM\IBM 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

IBM i On IBM i, the `mqs.ini` file is stored in `/QIBM/UserData/mqm`. The 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).

In particular, the `mqs.ini` file is used to locate the data associated with each queue manager.

Example mqs.ini file for AIX and Linux

```
Linux AIX
#*****#
#* 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

```

Example mqs.ini file for Windows

Windows

```

#*****#
#* 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=C:\ProgramData\IBM\MQ

LogDefaults:
  LogPrimaryFiles=3
  LogSecondaryFiles=2
  LogFilePages=4096
  LogType=CIRCULAR
  LogBufferPages=0
  LogDefaultPath=C:\ProgramData\IBM\MQ\log

QueueManager:
  Name=saturn.queue.manager
  Prefix=C:\ProgramData\IBM\MQ
  Directory=saturn!queue!manager
  InstallationName=Installation1

QueueManager:
  Name=pluto.queue.manager
  Prefix=C:\ProgramData\IBM\MQ
  Directory=pluto!queue!manager
  InstallationName=Installation2

DefaultQueueManager:
  Name=saturn.queue.manager

ApiExitTemplate:
  Name=OurPayrollQueueAuditor
  Sequence=2
  Function=EntryPoint
  Module=C:\usr\ABC\auditor

```

```
Data=123

ApiExitCommon:
  Name=MQPoliceman
  Sequence=1
  Function=EntryPoint
  Module=C:\usr\MQPolice\tmpq
  Data=CheckEverything
```

Example mqs.ini file for IBM i

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
```

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.

mqs.ini stanzas



Attention: This topic links to more information about the stanzas in the mqs.ini file. Each stanza contains information on the parameters in that stanza.

Multi Summary of mqs.ini file stanzas and attributes

A summary of the attributes of the stanzas of the IBM MQ configuration file, `mqs.ini`, with links to more information.

Table 10. Stanzas of the mqs.ini file	
Stanza and attributes	Description of attributes
AllQueueManagers stanza	
DefaultPrefix	The path to the <code>qmgrs</code> directory, within which the queue manager data is kept.
 DefaultEphemeralPrefix	The path to the directory, within which the queue manager ephemeral data is kept.
 ConvEBCDICNewline	How IBM MQ is to convert the EBCDIC NL character into ASCII format
ApiExitCommon stanza and ApiExitTemplate stanza	
Name	The descriptive name of the API exit passed to it in the <code>ExitInfoName</code> field of the MQAXP structure.
Function	The name of the function entry point into the module containing the API exit code.
Module	The module containing the API exit code.
Data	Data to be passed to the API exit in the <code>ExitData</code> field of the MQAXP structure.
Sequence	The sequence in which this API exit is called relative to other API exits.
DefaultQueueManager stanza	
Name	The name of the queue manager that processes any commands for which a queue manager name is not explicitly specified.
ExitProperties stanza	
CLWLMode	Whether the cluster workloac (CLWL) exit exit runs either in FAST mode or SAFE mode.
LogDefaults stanza	
LogPrimaryFiles	The log files allocated when the queue manager is created.
LogSecondaryFiles	The log files allocated when the primary files are exhausted.
LogFilePages	The number of log file pages. (The log file size is specified in units of 4 KB pages.)
LogType	The type of logging to be used by the queue manager (circular or linear).
LogBufferPages	The amount of memory allocated to buffer records for writing, specifying the size of the buffers in units of 4 KB pages.
LogDefaultPath	The directory in which the log files for a queue manager reside.

Table 10. Stanzas of the mqs.ini file (continued)

Stanza and attributes	Description of attributes
LogWriteIntegrity	The method the logger uses to reliably write log records.
QueueManager stanza	
Name	The name of the queue manager.
Prefix	Where the queue manager files are stored.
Directory	The name of the subdirectory under the prefix\QMGRS directory where the queue manager files are stored.
DataPath	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	The name of the IBM MQ installation associated with this queue manager.
EphemeralPrefix	Where the queue manager ephemeral data is stored.

Multi AllQueueManagers stanza of the mqs.ini file

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, and the method for converting EBCDIC-format data to ASCII format.

Use the AllQueueManagers stanza in the mqs . ini file to specify the information about all queue managers.

Windows Linux Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer General and Extended IBM MQ properties page.

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 AIX 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.

Multi DefaultEphemeralPrefix= directory_name

This attribute specifies the path to the directory, within which the queue manager ephemeral data is kept, such as IPC sockets, and is used only to set the **EphemeralPrefix** of a queue manager when a queue manager is created. In addition, you must create the directory yourself if you change the default value. You must create the ephemeral data directory with permissions that allow the IBM MQ group access to write to that directory.

As an alternative to changing the mqs . ini file, you can use the environment variable **MQ_EPHEMERAL_PREFIX** to override the **DefaultEphemeralPrefix** for the **crtmqm** command.

Because of operating system restrictions, the default ephemeral prefix is restricted to:

- **Linux** **AIX** 12 characters on AIX and Linux platforms.
- **IBM i** 24 characters on IBM i.

MQ Appliance **DefaultEphemeralPrefix** is not supported on the IBM MQ Appliance.

Multi **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.

IBM i On IBM MQ for IBM i, CCSID 1253 is considered to be an ISO CCSID, and NL_TO_LF affects both ISO and ASCII conversions.

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 11 on page 101](#).

<i>Table 11. List of possible ISO CCSIDs</i>	
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.

From IBM MQ 9.1.0 Fix Pack 2 and IBM MQ 9.1.2, 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.

ApiExitCommon and ApiExitTemplate stanzas of the mqs.ini file

The ApiExitTemplate and ApiExitCommon stanzas identify API exit routines for all queue managers.

Use the ApiExitTemplate and ApiExitCommon stanzas in the `mqs.ini` file to identify API exit routines for all queue managers. (To identify API exit routines for individual queue managers, you use the ApiExitLocal stanza, as described in “[ApiExitLocal stanza of the qm.ini file](#)” on page 120.)

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer Exits IBM MQ properties page.

 On Windows, you can also use the `amqmdain` command to change the entries for API exits.

For more information about using these attributes, see [Configuring API exits](#).

Name=ApiExit_name

The descriptive name of the API exit passed to it in the ExitInfoName field of the MQAXP structure.

This name must be unique, no longer than 48 characters, and contain only valid characters for the names of IBM MQ objects (for example, queue names).

Function=function_name

The name of the function entry point into the module containing the API exit code. This entry point is the MQ_INIT_EXIT function.

The length of this field is limited to MQ_EXIT_NAME_LENGTH.

Module=module_name

The module containing the API exit code.

If this field contains the full path name of the module it is used as is. If this field contains just the module name, the module is located using the **ExitsDefaultPath** attribute in the ExitPath stanza of the `qm.ini` file.

On platforms that support separate threaded libraries, you must provide both a non-threaded and a threaded version of the API exit module. The threaded version must have an `_t` suffix. The threaded version of the IBM MQ application stub implicitly appends `_t` to the given module name before it is loaded.

The length of this field is limited to the maximum path length the platform supports.

Data=data_name

Data to be passed to the API exit in the ExitData field of the MQAXP structure.

If you include this attribute, leading and trailing blanks are removed, the remaining string is truncated to 32 characters, and the result is passed to the exit. If you omit this attribute, the default value of 32 blanks is passed to the exit.

The maximum length of this field is 32 characters.

Sequence=sequence_number

The sequence in which this API exit is called relative to other API 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 contiguous. A sequence of 1, 2, 3 has the same result as a sequence of 7, 42, 1096. If two exits have the same sequence number, the queue manager decides which one to call first. You can tell which was called after the event by putting the time or a marker in ExitChainArea indicated by the ExitChainAreaPtr in MQAXP or by writing your own log file.

This attribute is an unsigned numeric value.

DefaultQueueManager stanza of the mqs.ini file

The DefaultQueueManager stanza specifies the default queue manager for the node.

Use the DefaultQueueManager stanza in the `mqs.ini` file to specify the default queue manager.

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer the General IBM MQ properties page.

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.

ExitProperties stanza of the mqs.ini file

The ExitProperties stanza specifies configuration options used by queue manager exit programs.

Use the ExitProperties stanza in the `mqs.ini` file to specify configuration options used by queue manager exit programs.

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer Extended IBM MQ properties page.

CLWLMode = SAFE (default) |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 (amqzlw0) 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 might 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.

LogDefaults stanza of the mqs.ini file

The LogDefaults stanza specifies information about log defaults for all queue managers.

Use the LogDefaults stanza in the `mqs.ini` file to specify information about log defaults for all queue managers.

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer Default log settings IBM MQ properties page.

If you require a non-default value, you must explicitly specify that value in the LogDefaults stanza.

If the LogDefaults stanza does not exist, then the IBM MQ defaults are 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. For more information about this command, see **crtmqm**.

Once a queue manager has been created, the log attributes for that queue manager are taken from the settings described in [“Log stanza of the qm.ini file” on page 142](#).

Note: 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 for the LogDefaults stanza are shown in [“Example mqs.ini file for AIX and Linux” on page 96](#) and [“Example mqs.ini file for Windows” on page 97](#). A value of zero for the LogBufferPages attribute means 512.

The default prefix, which is specified in the [“AllQueueManagers stanza of the mqs.ini file” on page 100](#), and log path specified for the particular queue manager, which is specified in the [“Log stanza of the qm.ini file” on page 142](#), 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 733](#).

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 (default) | 2-254 (Windows) | 2-510 (AIX and Linux)

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 AIX and Linux. The default is 3.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on AIX and Linux, 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 (default) | 1-253 (Windows) | 1-509 (AIX and Linux)

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 AIX and Linux. The default number is 2.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on AIX and Linux, 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 AIX and Linux, 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 (default) | 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 727 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 727 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 can change the logging method after a queue manager has been created. For more information, see [migmqlog](#).

LogBufferPages=0 (default) |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 which 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:

-  **Windows** *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 installation time.
-  **Linux**  **AIX** /var/mqm/log for AIX 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 `mqm.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 (default)

The method the logger uses to reliably write log records.

TripleWrite (default)

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`.

For more information, see [“LogWriteIntegrity - using SingleWrite or TripleWrite” on page 145](#).

QueueManager stanza of the mqs.ini file

The QueueManager stanza specifies the location of the queue manager directory.

There is one QueueManager stanza for every queue manager. The attributes of this stanza 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. For more information about name transformation, see [Understanding IBM MQ file names](#).

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](#) 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.

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.

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.

EphemeralPrefix= *name*

Where the queue manager ephemeral data is stored.

By default, this value is not present, meaning the data is stored under the Prefix location.

The value is set from the value of the **MQ_EPHEMERAL_PREFIX** environment variable, or the **DefaultEphemeralPrefix** attribute of the [AllQueueManagers](#) stanza in the `mqs.ini` file, when the queue manager is created.

 Because of operating system restrictions, the default ephemeral prefix is restricted to 24 characters on IBM i.

Related tasks

[“Associating a queue manager with an installation” on page 475](#)

When you create a queue manager, it is automatically associated with the installation that issued the **crtmqm** command. On AIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

Windows Advanced Configuration and Power Interface (ACPI)

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.

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.

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.

Queue manager configuration files, qm.ini

A queue manager configuration file, `qm.ini`, contains information relevant to a specific queue manager. The attributes that you can use to modify the configuration of an individual queue manager override any settings for IBM MQ.

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.

Note: For more information on how and when to edit a `qm.ini` file and on when any changes that you make to the file take effect, see [“Changing IBM MQ configuration information in .ini files on Multiplatforms”](#) on page 94.

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

Linux

AIX

On AIX 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
```

Windows

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 as follows:

```
C:\ProgramData\IBM\MQ\qmgrs\QMNAME\qm.ini
```

IBM i

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: You can change the *mqmdata directory* in the `mqs.ini` file.

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 [IBM MQ file names and Object names on IBM i](#).

qm.ini stanzas



Attention:

- This topic links to more information about the stanzas in the `qm.ini` file. Each stanza contains information on the parameters in that stanza, including an example where appropriate.
- Each stanza shows the platform, or platforms, of IBM MQ for Multiplatforms to which that stanza applies.

Multi

Automatic configuration of qm.ini at startup

You can configure your queue manager to automatically apply the contents of a file, or set of files, containing `qm.ini` overrides, on every queue manager start.

You can use this to have a configuration that can be modified and automatically replayed at the next queue manager restart. For example, if the `qm.ini` overrides are located on a mounted drive, it is possible to have a centralized configuration where the latest version is applied to every queue manager as they start.

You can use this functionality to simplify creating a uniform cluster, by using the automatic cluster functionality. For an example of this, see [“Creating a new uniform cluster”](#) on page 438.

Note: These overrides are only applied at queue manager start, and cannot influence the creation of the queue manager. For example, you cannot set the number of primary log files with this function.

Before you begin

You can use:

1. A single file, and create a text file containing changes to the `qm.ini` file.
2. A set of `qm.ini` format files:
 - To identify a directory where the configurations will exist, and
 - In that directory, create files, each with the extension `.ini`, for example, `qminisettings.ini`.

The file, or files, need only contain the stanza and **attribute=value** settings for the items that change. For example, to update the **MaxChannels** attribute in the Channels stanza, the file might contain:

```
Channels:
MaxChannels=1234
```

Note that in `qm.ini` override files, any line that is prefixed with `#` is treated as a comment.

Enabling automatic configuration of qm.ini file attributes

You can configure a new queue manager by using the **-ii** flag to the **crtmqm** command, and pointing either at a specific file, or directory. The supplied value is stored in the `qm.ini` file under the **AutoConfig** stanza, as attribute **IniConfig**.

You can configure an existing queue manager to enable automatic MQSC configuration, by adding the **AutoConfig** stanza attribute **IniConfig**, pointing to a valid file or directory. For example:

```
AutoConfig:
IniConfig=C:\MQ_Configuration\uniclus.ini
```

How does automatic configuration work?

During queue manager startup, the configuration that is identified by the **AutoConfig** stanza attribute **IniConfig** is validated, to ensure valid syntax, and then stored in the queue manager data tree into the `autocfg` directory as a single cached `.ini` file.

When multiple files from a directory are processed in, they are processed in alphabetical order.

During the first ever start of the queue manager, an inability to read the file or directory prevents the queue manager from starting, with an appropriate error message both to the console and to the queue manager error log.

On subsequent restarts, if the file or directory pointed to is unreadable, the previously cached file is used and a message written to the error log of the queue manager highlights this.

When you use the **strmqm** command, the contents of the cached `.ini` file are applied to the `qm.ini` file as overrides before the queue manager is invoked.

This does mean that for a standby queue manager, the settings are read when the **strmqm** command is processed, not when the queue manager becomes active.

How does the replacement qm.ini file get built?

The very first time that automatic initialization configuration is configured and the queue manager starts, a copy of the current `qm.ini` file is copied into the `autoconfig` subdirectory inside the data directory of the queue manager as `base_qm.ini`. This is considered the baseline from here onwards.

On every queue manager start, that is, **strmqm** time, the currently active `qm.ini` file is discarded and replaced with a copy of the `base_qm.ini`. Then the configuration from the `cached.ini` file is applied to this file.

Once a queue manager is under automatic configuration control, all changes to the `qm.ini` file should be performed through the file, or files, that are pointed to using the **IniConfig** attribute in the AutoConfig stanza.

Because the existing `qm.ini` file is removed at queue manager start, only the configuration in the supplied `qm.ini` file using the **IniConfig** attribute is applied to the queue manager base line.

If a stanza or attribute was changed through automatic initialization configuration on previous starts of the queue manager, those changes are removed unless they are still identified in the file or files identified by the **IniConfig** attribute.

Because of the recreation of the `qm.ini` file at queue manager start, this means any manual changes to the `qm.ini` file are lost. If you really need to make a change persistent, and cannot use the **IniConfig** attribute to make that change, you can do either of the following:

- Make the change to the `base_qm.ini` file itself.
- Delete the `base_qm.ini` file.

If you do delete this file, the `base_qm.ini` is re-created at the next queue manager start, based on the current contents of the `qm.ini` file. This *hardens* all the current changes as the new baseline for future starts.

Related concepts

[“Summary of qm.ini file stanzas and attributes” on page 110](#)

A summary of the attributes of the stanzas of the queue manager configuration file, `qmi.ini`, with links to more information.

Multi Summary of qm.ini file stanzas and attributes

A summary of the attributes of the stanzas of the queue manager configuration file, `qmi.ini`, with links to more information.

Table 12. Stanzas of the qm.ini file	
Stanza and attributes	Description of attributes
Windows AccessMode stanza	
Windows access group ¹	A Windows security group, members of which will be granted full access to all queue manager data files.
ApiExitLocal stanza	
Name	The descriptive name of the API exit passed to it in the ExitInfoName field of the MQAXP structure.
Function	The name of the function entry point into the module containing the API exit code.
Module	The module containing the API exit code.
Data	Data to be passed to the API exit in the ExitData field of the MQAXP structure.
Sequence	The sequence in which this API exit is called relative to other API exits.
Linux AIX AuthToken stanza	

Table 12. Stanzas of the *qm.ini* file (continued)

Stanza and attributes	Description of attributes
KeyStore	File path for the keystore that contains the trusted issuer's public key certificates or symmetric keys.
KeyStorePwdFile	File path for the file that contains the encrypted password for the keystore.
CertLabel	The certificate label for a public key certificate or symmetric key in the keystore that is used to validate authentication tokens.
UserClaim	Claim within the token that contains user identity information that the queue manager can adopt for authorization checks.
AllowOSGroups	This attribute determines whether the group membership for the adopted user is checked or not.
AutoCluster stanza	
Type	The type of automatic cluster. The only valid option is Uniform, which represents a uniform cluster.
ClusterName	The name of the automatic cluster.
RepositoryName1	The queue manager name for the first full repository in the automatic cluster.
Repository1Conname	The connection name (CONNAME) value for how members of the automatic cluster should connect to the queue manager.
RepositoryName2	The queue manager name for the second full repository in the automatic cluster.
Repository2Conname	The connection name (CONNAME) value for how members of the automatic cluster should connect to the queue manager.
AutoConfig stanza	
MQSCConfig	Either a full file path, or path to a directory, where all *.mqsc files are applied to the queue manager on every queue manager start.
IniConfig	Either a full file path, or path to a directory, where all *.ini file are applied to the qm.ini file on every queue manager start.
Channels stanza	
MaxChannels	The maximum number of current channels allowed.
MaxActiveChannels	The maximum number of channels allowed to be active at any time.
MaxInitiators	The maximum number of initiators.
MQIBindType	The binding for applications.
 ThreadedListener	Whether to start RUNMQLSR or AMQCLMAA as a listener.

Table 12. Stanzas of the *qm.ini* file (continued)

Stanza and attributes	Description of attributes
PipeLineLength	The maximum number of concurrent threads a channel will use.
AdoptNewMCA	Which types of channels can have the existing channel instance stopped so that a new channel instance can start when IBM MQ receives a request to start a channel, but finds that an instance of the channel is already running.
AdoptNewMCATimeout	The amount of time, in seconds, that the new channel instance waits for the old channel instance to end.
AdoptNewMCACheck	The type of checking required when enabling the AdoptNewMCA attribute.
ChlauthEarlyAdopt	The order in which connection authentication and channel authentication rules are processed.
PasswordProtection	Whether credentials specified by an application must be protected with MQCSP password protection, if the channel does not use TLS encryption.
IgnoreSeqNumberMismatch	Controls how the queue manager handles a sequence number mismatch during channel startup.
Connection stanza	
DefaultBindType	Whether applications and the queue manager, which run in separate processes, share some resources or no resources between them.
DiagnosticMessages stanza	
name	Name of a stanza.
Service	A service that is being enabled by this stanza.
ExcludeMessage	Messages that are not to be written to the queue manager error log.
SuppressMessage	Messages that are to be written to the queue manager error log once only in a specified time interval.
 SuppressInterval	The time interval, in seconds, in which messages specified in SuppressMessage are written to the queue manager error log once only.
Severities	A comma separated list of severity levels.
FilePath	The path to where the log files are written. (Only supported when the Service attribute is set to File.)
FilePrefix	The prefix of the log files. (Only supported when the Service attribute is set to File.)
FileSize	The size at which the log rolls over. (Only supported when the Service attribute is set to File.)
Format	The format of the file. (Only supported when the Service attribute is set to File.)

Table 12. Stanzas of the *qm.ini* file (continued)

Stanza and attributes	Description of attributes
  Syslog	The Syslog service that sends any unfiltered messages to syslog using the JSON format diagnostic messages specification.
  Ident	The ident value associated with the syslog entries. (Only supported when the Service attribute is set to Syslog.)
ExitPath stanza	
ExitsDefaultPath	The path for user exit programs on the queue manager system (32-bit).
ExitsDefaultPath64	The path for user exit programs on the queue manager system (64-bit).
ExitPropertiesLocal stanza	
CLWLMode	Whether the cluster workload (CLWL) exit runs either in FAST mode or SAFE mode.
   Filesystem stanza	
   ValidateAuth	Allow users who are not members of the mqm group to access error directories and files.
Log stanza	
LogPrimaryFiles	The log files allocated when the queue manager is created.
LogSecondaryFiles	The log files allocated when the primary files are exhausted.
LogFilePages	The number of log file pages. (The log file size is specified in units of 4 KB pages.)
LogType	The type of logging to be used by the queue manager (circular or linear).
LogBufferPages	The amount of memory allocated to buffer records for writing, specifying the size of the buffers in units of 4 KB pages.
LogPath	The directory in which the log files for a queue manager reside.
LogWriteIntegrity	The method the logger uses to reliably write log records.
LogManagement	The method used to manage log extents, either manually or by the queue manager.
 LU62 stanza	
 TPName	The TP name to start on the remote site.
 Library1	The name of the APPC DLL.
 Library2	The same as Library1, used if the code is stored in two separate libraries.

Table 12. Stanzas of the *qm.ini* file (continued)

Stanza and attributes	Description of attributes
CP4I NativeHAInstance stanza	
“Name” on page 146	The instance name that was used when the queue manager instance was created.
“ReplicationAddress” on page 146	The hostname, IPv4 dotted decimal, or IPv6 hexadecimal format address of the instance.
CP4I NativeHALocalInstance stanza	
“Name” on page 147	The name of the NativeHALocalInstance stanza, taken from the log replica instance name specified when the Native HA queue manager is created.
“KeyRepository” on page 147	The location of the key repository that holds the digital certificate to use for protection of log replication traffic.
“CertificateLabel” on page 147	The certificate label identifying the digital certificate to use for protection of log replication traffic.
“CipherSpec” on page 147	The MQ CipherSpec to use to protect log replication traffic.
“LocalAddress” on page 147	The local network interface address that accepts log replication traffic.
“HeartbeatInterval” on page 148	The heartbeat interval defines how often in milliseconds an active instance of a Native HA queue manager sends a network heartbeat.
“HeartbeatTimeout” on page 148	The heartbeat timeout defines how long a replica instance of a Native HA queue manager waits before it decides that the active instance is unresponsive.
“RetryInterval” on page 148	The retry interval defines how often in milliseconds a Native HA queue manager should retry a failed replication link.
Windows NETBIOS stanza	
Windows LocalName	The name by which this machine is known on the LAN.
Windows AdapterNum	The number of the LAN adapter.
Windows NumSess	The number of sessions to allocate.
Windows NumCmds	The number of commands to allocate.
Windows NumNames	The number of names to allocate.
Windows Library1	The name of the NetBIOS DLL.
QLErrorLog stanza	
ErrorLogSize	Specifies the size of the queue manager error log which it is copied to the backup.
ExcludeMessage	Specifies messages that are not to be written to the queue manager error log.

Table 12. Stanzas of the *qm.ini* file (continued)

Stanza and attributes	Description of attributes
SuppressMessage	Specifies messages that are written to the queue manager error log once only in a specified time interval.
SuppressInterval	Specifies the time interval, in seconds, in which messages specified in SuppressMessage are written to the queue manager error log once only.
Linux AIX Restricted Mode stanza ²	
Linux AIX ApplicationGroup	The name of the local transmission queue where remote messages are put if a transmission queue is not explicitly defined for their destination.
Security stanza	
ClusterQueueAccessControl	Check the access control of cluster queues or fully qualified queues hosted on cluster queue managers.
Windows GroupModel	Whether the Object Authority Manager (OAM) checks global groups when determining the group membership of a user on Windows.
z/OS DisablePolicyFIPSCheck	Allows Advanced Message Security (AMS) policies to be defined with algorithms that do not meet FIPS compliance if the queue manager is enabled for FIPS.
Service stanza	
Name	The name of the required service.
EntryPoints	The number of entry points defined for the service.
Windows SecurityPolicy	On Windows, the security policy for each queue manager
Linux AIX SecurityPolicy	On AIX and Linux, whether the queue manager uses user-based or group-based authorization. From IBM MQ 9.3.0, you can also create a non-operating system user name.
SharedBindingsUserId	For shared bindings only, whether the <code>UserIdentifier</code> field in the <code>IdentityContext</code> structure, from the <code>MQZ_AUTHENTICATE_USER</code> function, is the effective user ID or the real user ID.
FastpathBindingsUserId	For fastpath bindings only, whether the <code>UserIdentifier</code> field in the <code>IdentityContext</code> structure, from the <code>MQZ_AUTHENTICATE_USER</code> function, is the effective user ID or the real user ID.
IsolatedBindingsUserId	For isolated bindings only, whether the <code>UserIdentifier</code> field in the <code>IdentityContext</code> structure, from the <code>MQZ_AUTHENTICATE_USER</code> function, is the effective user ID or the real user ID.
Linux MQ Appliance AIX AllowLongUID	On AIX and Linux, whether to allow user IDs of up to 1024 characters (if the service specified is the default authorization service).
ServiceComponent stanza	

Table 12. Stanzas of the qm.ini file (continued)

Stanza and attributes	Description of attributes
Service	The name of the required service.
Name	The descriptive name of the service component.
Module	The name of the module to contain the code for this component.
ComponentDataSize	The size, in bytes, of the component data area passed to the component on each call.
Windows SPX stanza	
Windows Socket	The SPX socket number in hexadecimal notation.
Windows BoardNum	The LAN adapter number.
Windows KeepAlive	Switch the KeepAlive function on or off.
Windows Library1	The name of the SPX DLL.
Windows Library2	The same as LibraryName1, used if the code is stored in two separate libraries.
Windows ListenerBacklog	Override the default number of outstanding requests for the SPX listener.
SSL stanza	
 AllowedQuantumSafeAlgorithms	Specifies which quantum safe algorithms the queue manager supports as a TLS server or TLS client.
OutboundSNI	Specifies whether SNI capable clients will set SNI to the target IBM MQ channel name to the remote system when initiating a TLS connection or to the hostname.
AllowOutboundSNI	Specifies whether SNI capable clients will set SNI to the target IBM MQ channel name to the remote system when initiating a TLS connection.  Attention:  From IBM MQ 9.3.0 this property is deprecated. Use OutboundSNI instead.
AllowedCipherSpecs	Specifies a custom list of CipherSpecs that are ordered and enabled for use with IBM MQ channels on Multiplatforms.
AllowTLSV13	Whether a queue manager is able to use the TLS 1.3 CipherSpecs.
CDPCheckExtensions	Whether TLS channels on this queue manager try to check CDP servers that are named in CrlDistributionPoint certificate extensions.
 HTTPSCertRevocation	Sets the level of CRL/OCSP extension checks for HTTPS certificates.
 HTTPSCertValidation	Sets the level of certificate peer name validation performed when establishing HTTPS connections.

Table 12. Stanzas of the qm.ini file (continued)

Stanza and attributes	Description of attributes
“HTTPSKeyStore=string” on page 161	Provides the path to a pkcs12 key repository which can be used by the queue manager as a trust store when creating outgoing HTTPS connections, for example to a JWKS endpoint.
MinimumRSAKeySize	Specifies the minimum keysize that RSA certificates must have in order to be accepted.
OCSPAuthentication	The action to be taken when a revocation status cannot be determined from an OCSP server.
OCSPCheckExtensions	Whether TLS channels on this queue manager try to check OCSP servers that are named in AuthorityInfoAccess certificate extensions.
OCSPTimeout	The number of seconds to wait for an OCSP responder when performing a revocation check.
 PeerCertChainValidation	The IBM Global Security Kit (GSKit) certificate validation setting.
 RFC5280EKUChecks	Validates the Extended Key Usage (EKU) extension of the remote queue manager certificate where the local queue manager is acting as client.
SSLHTTPConnectTimeout	The number of seconds to wait for a network connection to be successfully established to a HTTP server when performing a revocation check.
SSLHTTPProxyName	Either the host name or network address of the HTTP Proxy server that is to be used by GSKit for OCSP checks.
Subpool stanza “3” on page 119	This stanza is created by IBM MQ. Do not change it .
ShortSubpoolName “3” on page 119	A name corresponding to a directory and symbolic link created inside the /var/mqm/sockets directory, which IBM MQ uses for internal communications between its running processes.
 TCP stanza	
Port	The default port number, in decimal notation, for TCP/IP sessions.
 Library1	The name of the TCP/IP sockets DLL.
 SecureCommsOnly	Specifies whether plain text communication is allowed, the default value, or not allowed.
KeepAlive	Switch the KeepAlive function on or off.
ListenerBacklog	Override the default number of outstanding requests for the TCP/IP listener.
Connect_Timeout	The number of seconds before an attempt to connect the socket times out.
SndBuffSize	The size in bytes of the TCP/IP send buffer used by the sending end of channels.

Table 12. Stanzas of the *qm.ini* file (continued)

Stanza and attributes	Description of attributes
RcvBuffSize	The size in bytes of the TCP/IP receive buffer used by the receiving end of channels.
RcvSndBuffSize	The size in bytes of the TCP/IP send buffer used by the sender end of a receiver channel.
RcvRcvBuffSize	The size in bytes of the TCP/IP receive buffer used by the receiving end of a receiver channel.
SvrSndBuffSize	The size in bytes of the TCP/IP send buffer used by the server end of a client-connection server-connection channel.
SvrRcvBuffSize	The size in bytes of the TCP/IP receive buffer used by the server end of a client-connection server-connection channel.
 DNSResolutionOrder	The order in which IBM MQ looks up host names in mixed IPv6 and IPv4 environments.
Tuning parameters stanza	
SuppressDspAuthFail	Whether the queue manager suppresses generation of authorization events and the writing of AMQ8077 error messages to the error log when an authorization check fails, if the connection lacks +dsp authority to an object.
ImplSyncOpenOutput	The minimum number of applications that have the queue open for put, before an implicit syncpoint might be enabled for a persistent put, outside of syncpoint.
UniformClusterName	The name of the IBM MQ cluster you are using as a uniform cluster.
OAMLdapConnectTimeout	The maximum time, in seconds, that the LDAP client will wait to establish a TCP connection to the server.
OAMLdapQueryTimeLimit	The maximum time, in seconds, that the LDAP client will wait to receive a response to an LDAP request from the server.
OAMLdapResponseWarningTime	If a connection to an LDAP server took longer than the threshold number of seconds specified by the OAMLdapResponseWarningTime parameter, then an AMQ5544W message will be written to the error log.
ExpiryInterval	Indicates the frequency with which the queue manager scans the queues looking for expired messages that have not already been cleaned up by other queue activities. It is a time interval in seconds.
LivenessHeartBeatLen	Configures the frequency with which the queue manager checks that writes to the log are being made at a reasonable rate.
ECHeartBeatLen	Configures the frequency of the queue manager general health checks.

Table 12. Stanzas of the qm.ini file (continued)	
Stanza and attributes	Description of attributes
<u>FileLockHeartBeatLen</u>	Changes the default value for the file lock checks for a multi-instance queue manager that the execution controller periodically performs to make sure that it still holds the exclusive lock on the primary multi-instance file.
Variables stanza	
<u>attribute=value</u>	A name and associated value for use as an insert during MQSC definitions.
XAResourceManager stanza	
<u>Name</u>	The resource manager instance.
<u>SwitchFile</u>	The fully-qualified name of the load file containing the resource manager's XA switch structure.
<u>XAOpenString</u>	The string of data to be passed to the resource manager's xa_open entry point.
<u>XACloseString</u>	The string of data to be passed to the resource manager's xa_close entry point.
<u>ThreadOfControl</u>	The value that the queue manager uses for serialization when it needs to call the resource manager from one of its own multithreaded processes. Mandatory for Windows.

Notes:

1. 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.
2. 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.
3. The Subpool stanza, and the 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.

AccessMode stanza of the qm.ini file

Access Mode applies to Windows servers only. The AccessMode stanza of the qm.ini file 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.

Example stanza

```
AccessMode:  
SecurityGroup=wmq\wmq
```

Related concepts

[“Secure unshared queue manager data and log directories and files on Windows” on page 572](#)

[“Securing shared queue manager data and log directories and files on Windows” on page 569](#)

Related tasks

[“Creating a multi-instance queue manager on domain workstations or servers on Windows” on page 545](#)

Related reference

[crtmqm \(create queue manager\)](#)

ApiExitLocal stanza of the qm.ini file

The `ApiExitLocal` stanza specifies API exit routines for a queue manager.

For a server, modify the `ApiExitLocal` stanza of the `qm.ini` file to identify API exit routines for a queue manager.

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer Exits queue manager properties page.

For a client, modify the `ApiExitLocal` stanza in the `mqclient.ini` file to identify API exit routines for a queue manager.

Overview

The `ApiExitLocal` stanza allows only a single `Module` to be specified, and yet four modules need to be provided, as follows:

- 32 bit unthreaded
- 32 bit threaded
- 64 bit unthreaded

- 64 bit threaded

Note that IBM MQ appends `_r` to the supplied module name to identify the threaded version of the exit, but IBM MQ does not provide a directly equivalent mechanism for the 32 bit and 64 bit variants.

The versions of `amqsaxe0` and `amqsaxe0_r` that are shipped in *prefix/mqm/samp/bin* are built for the native size of the queue manager on the platform for which they are built (now all 64 bit) and can only be used by applications running in the same native size.

If an unqualified module name is provided, IBM MQ looks in `/var/mqm/exits` for the 32 bit variants and in `/var/mqm/exits64` for the 64 bit variants

For example, `module=amqsaxe` implies:

```
/var/mqm/exits/amqsaxe - 32 bit unthreaded variant
/var/mqm/exits/amqsaxe_r - 32 bit threaded variant
/var/mqm/exits64/amqsaxe - 64 bit unthreaded variant
/var/mqm/exits64/amqsaxe_r - 64 bit threaded variant
```

Windows 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 [“ApiExitCommon and ApiExitTemplate stanzas of the mq.ini file”](#) 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 more information about using these attributes, see [Configuring API exits](#).

Parameters

Name=ApiExit_name

The descriptive name of the API exit passed to it in the `ExitInfoName` field of the `MQAXP` structure.

This name must be unique, no longer than 48 characters, and contain only valid characters for the names of IBM MQ objects (for example, queue names).

Function=function_name

The name of the function entry point into the module containing the API exit code. This entry point is the `MQ_INIT_EXIT` function.

The length of this field is limited to `MQ_EXIT_NAME_LENGTH`.

Module=module_name

The module containing the API exit code.

If this field contains the full path name of the module it is used as is. If this field contains just the module name, the module is located using the **ExitsDefaultPath** attribute in the `ExitPath` stanza of the `qm.ini` file.

On platforms that support separate threaded libraries, you must provide both a non-threaded and a threaded version of the API exit module. The threaded version must have an `_r` suffix. The threaded version of the IBM MQ application stub implicitly appends `_r` to the given module name before it is loaded.

The length of this field is limited to the maximum path length the platform supports.

Data=data_name

Data to be passed to the API exit in the `ExitData` field of the `MQAXP` structure.

If you include this attribute, leading and trailing blanks are removed, the remaining string is truncated to 32 characters, and the result is passed to the exit. If you omit this attribute, the default value of 32 blanks is passed to the exit.

The maximum length of this field is 32 characters.

Sequence=sequence_number

The sequence in which this API exit is called relative to other API 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 contiguous. A sequence of 1, 2, 3 has the same result as a sequence of 7, 42, 1096. If two exits have the same sequence number, the queue manager decides which one to call first. You can tell which was called after the event by putting the time or a marker in ExitChainArea indicated by the ExitChainAreaPtr in MQAXP or by writing your own log file.

This attribute is an unsigned numeric value.

Example stanza

```
ApiExitLocal:
  Name=ClientApplicationAPIChecker
  Sequence=3
  Function=EntryPoint
  Module=/usr/Dev/ClientAppChecker
  Data=9.20.176.20
```

AuthToken stanza of the qm.ini file

Use the **AuthToken** stanza to configure the queue manager to validate authentication tokens that are provided by connecting applications. If your authentication service supports a JWKS endpoint for key configuration this is usually a preferable option.

See [Configuring a queue manager to accept authentication tokens using a JWKS endpoint](#) for more information.

The AuthToken stanza

KeyStore=string

File path for the keystore that contains the trusted issuer's public key certificates and symmetric keys. You can add the keys to an existing keystore or create a new keystore. For more information, see [Configuring a queue manager to accept authentication tokens](#). The queue manager uses the keys in the keystore to verify that the authentication token that the application presents is signed by the trusted issuer.

You can use either a CMS keystore with the file extension .kdb or a PKCS#12 keystore with the file extension .p12. If the keystore file does not exist or it cannot be accessed, then an AMQ7076E: Invalid value for attribute in ini file error is output to the queue manager's error log.

Ensure that the keystore type matches the file name extension for the keystore. IBM MQ detects the correct format of the keystore, inconsistencies might cause other administrative issues however, if the keystore type and file name extension do not match.

The maximum length of the keystore file path is 256 characters.

KeyStorePwdFile=string

File path for the file that contains the encrypted password for the keystore. The file must contain the encrypted password as a single line of text. Plain text passwords are not accepted.

Use the **runqmcrcd** command to encrypt the password before you save it in the keystore password file. The keystore password file must contain only the encrypted password that is created by running the **runqmcrcd** command.

The maximum length of the plain text password before it is encrypted is 1024 characters.

This parameter is optional. If it is not provided, the queue manager looks for a stash file with the password in the same directory and with the same name as the keystore, but with the file extension .sth. If the stash file is not found, the configuration is rejected and error message AMQ7006E is output to the queue manager's error log. For more information about the options to store keystore passwords, see [Encrypting key repository passwords](#).

The maximum length of the password file path is 256 characters.

CertLabel=string

The certificate label for a public key certificate or symmetric key in the keystore that is used to validate authentication tokens. You can provide up to 32 certificate labels by repeating the **CertLabel** attribute.

When you add certificates to the queue manager keystore, give them meaningful labels. Certificate labels are case-sensitive. They can contain alphanumeric characters, punctuation characters, and spaces. If an invalid character is detected, an error is returned and an error message is written to the IBM MQ error log.

Trusted token issuers might provide multiple public key certificates and symmetric keys. For example, public key certificates have validity periods. When they are close to expiry, the token issuer provides a new certificate with a fresh expiry date. For a time, both certificates might be valid.

When applications present tokens for authentication, the list of **CertLabels** is checked until a valid key that was used to sign the token is found. If the match is found, the token signature is validated.

If **CertLabel** is not specified, the connection from the application that presents the token fails with reason code 2063 MQRC_SECURITY_ERROR, and message AMQ5786E: Authentication token configuration error is written to the queue manager's error log.

The maximum length of the certificate label is 64 characters.

For example,

```
AuthToken:
  KeyStore=/var/mqm/qmgrs/qmgrs/qm1/tokenissuer/key.kdb
  KeyStorePwdFile=/var/mqm/qmgrs/qm1/tokenissuer/key.pw
  CertLabel=token
  CertLabel=rsakey
  CertLabel=mark
  ... up to 32 CertLabel fields
```

UserClaim=string

Claim within the token that contains the user ID that the queue manager adopts for authorization checks.

This parameter is optional if the queue manager is configured with **ADOPTCTX(NO)**. If **ADOPTCTX(YES)** is used, then this parameter is required. **ADOPTCTX** is an attribute present in the authentication information (AUTHINFO) object that is referenced by the queue manager's **CONNAUTH** attribute.

To adopt an identity, the token must contain a claim with the name that is specified in the **UserClaim** attribute of the **AuthToken** stanza and **ADOPTCTX(YES)** must be used.

For example, if your token includes a claim "AppUser": "MyUserName", then you must specify **UserClaim=AppUser** in the AuthToken stanza of the **qm.ini** file to adopt the identity "MyUserName" for authorization.

The maximum length of the **UserClaim** attribute value is 128 characters.

Note: From IBM MQ 9.4.0, if the AuthToken stanza is specified, the effective value of the **SecurityPolicy** attribute of the Service stanza is set to **UserExternal**. Token authentication is not available if **SecurityPolicy** is explicitly set to **Group** in the Service stanza. If **SecurityPolicy** is set to **Group**, remove the **SecurityPolicy** attribute from the Service stanza, then restart the queue manager. For more information, see [SecurityPolicy](#).

Note: Use the **ADOPTCTX** attribute of the authentication information object to control whether the user ID in the token is adopted for authorization checks. When you create the queue manager, this attribute is set to **ADOPTCTX(YES)**. This value causes the user ID from the token to be adopted. The user ID must meet the requirements for user IDs in authentication tokens. For more information, see [User IDs in authentication tokens](#). If the token user claim contains a user ID that does not meet the requirements, the connection is rejected with reason code **2035 MQRC_NOT_AUTHORIZED**. If **ADOPTCTX(NO)** is set, the token is used only for authentication, and another user must be used for authorization.

AllowOSGroups=NO (default) |YES

The default value is NO. Determines whether an identity that is adopted from a token is treated as an operating system (OS) user, and whether the group memberships of the matching OS user are honored during authorization. .

AllowOSGroups=NO|N

Authorization checks are based only on the name of the user that is adopted from the token.

AllowOSGroups=YES|Y

Authorization checks are based on the name of the user and the groups they might belong to are also checked.

Note: The **allowOSGroups** attribute of the **AuthToken** stanza still takes effect throughout token authentication, even if all other token validation configuration is managed through the **JWKS** stanza.

Example stanza - authentication only

It is valid for this stanza to contain solely the **AllowOSGroups** attribute. However, if local keystore configuration is being included, then it must contain as a minimum:

- **KeyStore** file path, and
- **CertLabel** name.

```
AuthToken:
  KeyStore=/var/mqm/qmgrs/qmgrs/qm1/tokenissuer/key.kdb
  CertLabel=token
  ... up to 32 CertLabel fields
```

If you only included the two minimum parameters, then:

- A stash file `key.sth` must exist with the encrypted keystore password so the keystore password file is not required.
- The token does not contain a username that is to be passed to IBM MQ for authorization. The application can connect and be authenticated but a different mechanism must be in place to provide authorization for the application to do work after it is connected.

Depending on the configuration for your queue manager, the username that is used for authorization might be the one defined on the channel through MCA rules or the username that the client app ran as might exist on your server and belong to groups with authorities. Keep in mind that when you use tokens:

- Your queue manager is put into **UserExternal** mode, which means users that do not exist on the operating system where the queue manager is running can be used for authentication.
- Even if you do not include the **AllowOSGroups** option in your **AuthToken** `qm.ini` stanza, the default is set to No. Therefore, if you include the **UserClaim** but do not specify the **AllowOSGroups=Yes**, the token user that is adopted for authorization is not checked for groups that it might belong to on the operating system where the queue manager is running.

Example stanza - authentication and authorization

You can define all the **AuthToken** parameters:

- **KeyStore** file path,
- **KeyStorePwdFile** file path,
- **CertLabel** name,
- **UserClaim** name and
- **AllowOSGroups** option.

```
AuthToken:
  KeyStore=/var/mqm/qmgrs/qmgrs/QMJWT/ssl/key.kdb
  KeyStorePwdFile=/var/mqm/qmgrs/QMJWT/ssl/key.pw
  CertLabel=token
  CertLabel=rsaKey
```



```
CertLabel=mark
... up to 32 CertLabel fields
UserClaim=AppUser
AllowOSGroups=Y
```

If you included all the available parameters, then:

- Encrypt the password for the keystore by using the **runqmcrcd** command. Save it in a file then include the file path in the **AuthToken** stanza.
- The username that is in the authentication token user claim is used for both authentication and authorization.
 - The token user might exist as a user on the operating system where the queue manager is running.
 - You defined an authentication information object to enable user checking.
 - You set up channel authentication records to adopt a user with authorization to interact with IBM MQ objects, based on the channel authentication or MCA rules.

Your strategy for authenticating and authorizing token users depends on your requirements and how your IBM MQ queue managers are already configured. For more information, see the [Working with authentication tokens](#).

Related concepts

[Working with tokens](#)

Related tasks

[Configuring a queue manager to accept AuthTokens](#)

[Using authentication tokens in an application](#)

AutoCluster stanza of the qm.ini file

The AutoCluster stanza is used when the queue manager starts to identify if the cluster is a member of an automatic cluster, and can identify the full repositories of the cluster.

The following attributes are mandatory for the AutoCluster stanza:

Type=*Uniform*

Specifies the type of automatic cluster, and the only valid option is *Uniform*, which represents a uniform cluster.

ClusterName=<*String*>

The name of the cluster, that is the automatic cluster name.

The following attributes are optional for the AutoCluster stanza but you must provide them in pairs:

RepositoryName1=<*String*>

This is the queue manager name for the first full repository in the automatic cluster. This can be the name of this queue manager, or another one.

Repository1Conname=<*Connection Name String*>

This is the connection name (CONNAME) value for how members of the automatic cluster should connect to this queue manager.

Repository2Name=<*String*>

This is the queue manager name for the second full repository in the automatic cluster. This can be the name of this queue manager, or another one.

Repository2Conname=<*Connection name string*>

This is the connection name (CONNAME) value for how members of the automatic cluster should connect to this queue manager.

Example stanza

```
AutoCluster:
  Repository1Name=QM1
  Repository2Name=QM2
  Repository1Conname=127.0.0.1(1414)
```

```
Repository2Conname=127.0.0.1(1415)
ClusterName=UNIFORMCLUSTER1
Type=Uniform
```

Related concepts

[“Automatic application balancing” on page 425](#)

Automatic application balancing greatly enhances application distribution and availability by enabling an IBM MQ uniform cluster to closely manage application distribution across the cluster, rather than relying on randomization, or a manual pinning of applications to specific queue managers.

Related tasks

[“Creating a new uniform cluster” on page 438](#)

How you create a new uniform cluster.

Related reference

[“Using automatic cluster configuration” on page 442](#)

You configure IBM MQ to enable automatic configuration by changing the `qm.ini` configuration information.

AutoConfig stanza of the qm.ini file

The attributes of the AutoConfig stanza are frequently used as part of setting up uniform clusters.

Note: You can only use the AutoConfig stanza for uniform clusters.

MQSCConfig=<Path>

The path is either a full file path, or path to a directory, where all `*.mqsc` files are applied to the queue manager on every queue manager start.

For more information, see [Automatic configuration from an MQSC script at startup](#).

IniConfig=<Path>

The path is either a full file path, or path to a directory, where all `*.ini` files are applied to the `qm.ini` file on every queue manager start.

For more information, see [“Automatic configuration of qm.ini at startup” on page 108](#).

ConfigTimeout

The value (in seconds) that the queue manager waits for auto configuration to complete. After that time, the queue manager continues to startup and is available for applications to connect.

The default behavior is no timeout. This means that the queue manager is not available for applications to connect, until all of the auto configuration commands have been completed.

You should not configure this attribute simply because configuration is taking a long time, because applications might be able to connect before configuration that applies to them has been completed, for example, the creation of queues needed by the application.

Example stanza

```
AutoConfig:
  MQSCConfig=/tmp/auto.mqsc
  IniConfig=/tmp/auto.ini
  ConfigTimeout=120
```

Related concepts

[“Automatic application balancing” on page 425](#)

Automatic application balancing greatly enhances application distribution and availability by enabling an IBM MQ uniform cluster to closely manage application distribution across the cluster, rather than relying on randomization, or a manual pinning of applications to specific queue managers.

Related tasks

[“Creating a new uniform cluster” on page 438](#)

How you create a new uniform cluster.

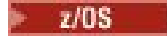
Related reference

[“Using automatic cluster configuration” on page 442](#)

You configure IBM MQ to enable automatic configuration by changing the `qm.ini` configuration information.

Channels stanza of the `qm.ini` file

The attributes of the Channels stanza determine the configuration of a channel.

 This information is not applicable to IBM MQ for z/OS.

Use the CHANNELS stanza in the `qm.ini` file to specify information about channels.

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer Channels queue manager properties page.

MaxChannels = 100 (default) |number

The maximum number of *current* channels allowed. For information about channel states, and what a *current* channel is, see [“Channel states” on page 244](#).

The default is 100.

You can set **MaxChannels** to a different value to limit the maximum number of current channels if required. For IBM MQ Appliance, the default value is 999 999 999.

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 (default) |number

The maximum number of initiators. The default and maximum value is 3.

MQBindType=FASTPATH|STANDARD

The binding for applications:

FASTPATH

Channels connect using MQCONN FASTPATH; 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.

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.

See [Multiple thread support - pipelining](#) for more information.

AdoptNewMCA = NO (default) |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 (default)|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.

ChlauthEarlyAdopt = Y (default) | N

The order in which connection authentication and channel authentication rules are processed is a significant factor in determining the security context for IBM MQ client application connections.



Attention: The default if **ChlauthEarlyAdopt** is not present in the qm.ini file is N. However, all queue managers are created with **ChlauthEarlyAdopt=Y** automatically added to the qm.ini file.

ChlauthEarlyAdopt only adopts user IDs that have been provided to a queue manager for connection authentication, if **ADOPTCTX(YES)** is set on the connection authentication AUTHINFO object on the queue manager.

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.

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 (default) |always|optional|warn

Authentication credentials that IBM MQ client applications specify when they connect to a queue manager can be protected by using the IBM MQ MQCSP password protection feature, if the connection does not use TLS encryption.

MQCSP password protection is useful for test and development purposes as using MQCSP password protection is simpler than setting up TLS encryption, but not as secure.

For more information about protecting credentials in the MQCSP structure, and the values that can be set for this attribute, see [MQCSP password protection](#).

IgnoreSeqNumberMismatch = NO (default) | YES

The Message Channel Agents (MCAs) at the two ends of a channel each keep count of the number of messages sent through the channel to maintain synchronization. Synchronization can be lost, for example if the channel definition at one end is deleted and then re-created. Under these circumstances a **RESET CHANNEL** may be required to acknowledge that synchronization data has been lost and permit the channel to continue startup.

The **IgnoreSeqNumberMismatch** attribute must be set on the receiver queue manager.

Effectively, this attribute performs a reset channel command on the receiver channel.

This attribute controls how the queue manager handles a sequence number mismatch during channel startup using the following values:

NO

Channel sequence numbers are checked during channel resynchronization, if the two MCAs do not agree on the same sequence number, error message AMQ9526 will be reported and the channel will fail to start.

YES

Channel sequence numbers are checked during channel resynchronization, but if the two MCAs do not agree on the same sequence number, warning message AMQ9703 will be reported and the channel startup will continue. This attribute value should not be needed under normal circumstances. When it is known that synchronization data has been lost, for example during disaster recovery, this option avoids needing to manually acknowledge each sequence number mismatch. Specifying this value has a similar effect to an administrator automatically issuing a **RESET CHANNEL** in response to each sequence number mismatch.

ChlauthIgnoreUserCase = N (default) | Y

Enables a queue manager to make username matching within CHLAUTH rules case-insensitive. This option allows the:

- CLNTUSER in CHLAUTH TYPE(USERMAP) rules to be matched case-insensitively
- USERLIST in CHLAUTH TYPE(BLOCKUSER) rules to be matched case-insensitively

Valid values for **ChlauthIgnoreUserCase** are the following values:

N

The channel authentication rules attempt to match client user identification with case sensitivity, for example a rule specifying CLNTUSER('Fred') will not match 'fred' or 'FRED', it will only match a user identifier of 'Fred'. This is the default value.

Y

The channel authentication rules attempt to match client user identification with case insensitivity, for example a channel authentication rule with TYPE(USERMAP) or TYPE(USERBLOCK) specifying CLNTUSER('Fred') will match with any variation of case, for example user identifiers 'Fred', 'FRED' and 'fred' all match.

Note that, when ignoring case of user identifiers when matching channel authentication rules, it is possible for more than one rule to match. If this occurs, the rule that is matched is undefined. For example with the following rules, if user 'fred' connects to a queue manager via the CLIENT channel, they might be mapped to 'mquser1' or 'mquser2':

```
SET CHLAUTH('CLIENT') TYPE(USERMAP) CLNTUSER('fred') USERSRC(MAP) MCAUSER('mquser1')
SET CHLAUTH('CLIENT') TYPE(USERMAP) CLNTUSER('FRED') USERSRC(MAP) MCAUSER('mquser2')
```

To avoid any uncertainty when using `ChlauthIgnoreUserCase=Y`, avoid defining CHLAUTH rules that would overlap and result in different behavior when using a case-insensitive match.

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.

Example stanza

```
Channels:
  MaxChannels=200
  MaxActiveChannels=100
  MQIBindType=STANDARD
  PipelineLength=2
```

Related concepts

[“Channel states” on page 244](#)

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.

Connection stanza of the qm.ini file

The Connection stanza defines the default bind type.

Use the Connection stanza in the `qm.ini` file to specify the default bind type.

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer Extended queue manager properties page.

Note: You must create a Connection stanza if you need one.

DefaultBindType = SHARED (default) | 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 MQCONN with MQCNO_STANDARD_BINDING.

Changing the **DefaultBindType** might cause some applications to degrade in performance.

Example stanza

```
Connection:
  DefaultBindType=SHARED
```

Multi Diagnostic message logging

The diagnostic message logs of IBM MQ are a mechanism, to allow various components of the IBM MQ system, to report diagnostic messages relating to IBM MQ configuration and runtime state changes and issues.

These logs are sometimes referred to as IBM MQ *error logs*, but have always contained IBM MQ information and warning messages, as well as error messages. The three primary components of IBM MQ which report to these logs are:

- Queue Managers
- IBM MQ Clients
- The remainder of the IBM MQ system

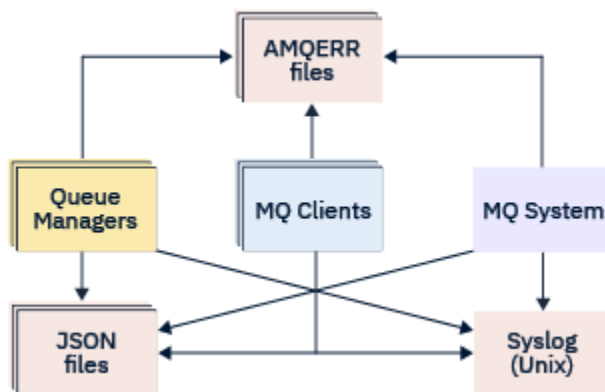
IBM MQ supports reporting diagnostics messages through a number of different methods known as *diagnostics message services*, allowing for a tailored approach to recording and consuming this information:

- AMQERRnn log files
- JSON formatted log files
- **AIX** Syslog in JSON format

JSON output by IBM MQ is formatted as single line JSON objects, such that each individual line of the JSON log, or Syslog record, represents a valid JSON object. The log as a whole is not encapsulated as a single JSON object.

The following illustration shows that queue managers, IBM MQ clients, and the IBM MQ system can *all* report diagnostic messages using the methods described.

Figure 5. How different parts of IBM MQ can report diagnostic messages



How IBM MQ diagnostics logs are configured:

Diagnostic logs are defined and customized using stanzas within the `qm.ini` file particular to the IBM MQ component that requires them. Each unique logging end point is defined under its own stanza heading within the ini file, along with any customizations defined within it. Customizations can include:

- The size of log files to wrap on, before rollover will occur; not applicable to Syslog
- Any filtering based on the severity of the log messages, and
- Any specific message codes to suppress.

IBM MQ can be configured to write to any, or all, of the three types of logging end points, allowing for particular log stanzas to fulfill particular roles. Similarly, multiple file services can be defined. For example:

- JSON format makes for easier parsing through automated tooling in local and Cloud environments.

- Syslog output allows IBM MQ components to integrate diagnostics information into a common OS logging location in line with other products on the system.
- Log end points filtered based on severity allowing for particular log files to record, for example, only severe errors in the system.

Regardless of the style of diagnostics logging configured, the traditional diagnostics files held under the IBM MQ system log directory (`/var/mqm/errors/AMQERRnn.log`) and specific queue manager log directory (`/var/mqm/qmgrs/<qmgr_name>/errors/AMQERRnn.log`) are always written to, in addition to any other logging configuration used.

For queue managers only, optional configuration of these mandatory logs can be performed by specifying attributes of the [“Diagnostic message service stanzas”](#) on page 135.

Different stanza areas

The additional stanzas can be applied to different areas of IBM MQ.

Qmgr(qm.ini)

Applies to log message generated by the queue manager

System(mqs.ini)

Applies to log messages generated by the system. This option is not specific to a queue manager, except when a queue manager cannot access or write to its own logs.

Templates(mqs.ini)

One or more stanzas as templates, which are copied to `qm.ini` when a queue manager is created.

Client(mqcclient.ini)

Applies to client operation, for example `runmqsc` in client mode to a remote queue manager.

Converting between JSON formatted and traditionally formatted logs

The `mqrcc` command has been enhanced to allow a number of conversions between JSON and traditionally formatted logs, and between different languages.

Related reference

[“Diagnostic message service stanzas”](#) on page 135

The diagnostic message service options available enable customization of your IBM MQ diagnostics logging, so that log output can be directed to different log end points from different components of IBM MQ.

[“QMErrorLog stanza”](#) on page 133

You use the queue manager error log stanza `QMErrorLog` in the `qm.ini` file to tailor the operation and contents of IBM MQ error logs.

[“Diagnostic message services”](#) on page 138

The following diagnostic message services and their service specific attributes, specified under the `DiagnosticSystemMessages`, `DiagnosticMessages` and `DiagnosticMessagesTemplate` stanzas of your configuration files can be defined:

QMErrorLog stanza

You use the queue manager error log stanza `QMErrorLog` in the `qm.ini` file to tailor the operation and contents of IBM MQ error logs.

The `QMErrorLog` service is the traditional IBM MQ diagnostics logging service used to output diagnostics messages pertaining to the queue manager. The `QMErrorLog` service runs continuously and cannot be turned off, but can be customized to some extent.

You can use the `QMErrorLog` stanza in the `qm.ini` file to exclude certain messages from being written to the queue manager error log. You can also suppress messages from being written to the error log for a given interval of time.

Windows **Linux** Alternatively, instead of editing the `qm.ini` file directly, you can use the [Extended Queue Manager properties page](#) in IBM MQ Explorer to exclude and suppress messages with the **Excluded messages**, **Suppressed messages** and **Suppressed messages interval** attributes.



Attention:

- **Windows** You can use IBM MQ Explorer to make the changes only if you are using a local queue manager on the Windows platform.
- The QMErrorLog stanza is not applicable to the IBM MQ system configuration file, `mqs.ini`, or the client configuration file, generally named `mqclient.ini`.

The following attributes can be included in the QMErrorLog stanza:

ErrorLogSize= *maxsize*

Specifies the size of the queue manager error log that is copied to the backup. *maxsize* must be in the range 32768 through 2147483648 bytes. If **ErrorLogSize** is not specified, the default value of 33554432 bytes (32 MB) is used.

You can use this attribute to reduce the maximum size back to the previous maximum of 2 MB, if required.

You can set the size of the log using the **MQMAXERRORLOGSIZE** environment variable.

ExcludeMessage= *msgIds*

Specifies messages that are not to be written to the queue manager error log.

See `ExcludeMessage` in [“Diagnostic message service stanzas”](#) on page 135 for more information.

SuppressMessage= *msgIds*

Specifies messages that are written to the queue manager error log once only in a specified time interval. If the same message ID is specified in both `SuppressMessage` and `ExcludeMessage`, the message is excluded.

This option is not applicable to diagnostic message services defined in `mqclient.ini`. For more information, see `SuppressMessage` in [“Diagnostic message service stanzas”](#) on page 135.

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.

Example stanza

```
QMErrorLog:
  ErrorLogSize=262144
  ExcludeMessage=7234
  SuppressMessage=9001,9002,9202
  SuppressInterval=30
```

Related concepts

[“Queue manager configuration files, qm.ini”](#) on page 108

A queue manager configuration file, `qm.ini`, contains information relevant to a specific queue manager. The attributes that you can use to modify the configuration of an individual queue manager override any settings for IBM MQ.

Related reference

[“Diagnostic message service stanzas”](#) on page 135

The diagnostic message service options available enable customization of your IBM MQ diagnostics logging, so that log output can be directed to different log end points from different components of IBM MQ.

Diagnostic message service stanzas

The diagnostic message service options available enable customization of your IBM MQ diagnostics logging, so that log output can be directed to different log end points from different components of IBM MQ.

You enable additional diagnostic message services, using a stanza with one of the following names:

- **DiagnosticSystemMessages**

Defines the services used when a diagnostic message is generated that goes to the system error log. Valid in the `mqs.ini` or `mqclient.ini` files.

Client applications use a **DiagnosticSystemMessages** stanza in the `mqclient.ini` file and in `mqs.ini`, the **DiagnosticSystemMessages** stanza controls messages for a server application that does not have a queue manager context.

It is possible for you to configure a queue manager and applications that additionally write all messages to the syslog service.

- **DiagnosticMessages**

Defines the services used when a diagnostic message is generated that goes to the queue manager error log. Valid only in the `qm.ini` file.

- **DiagnosticMessagesTemplate**

A stanza that is copied from the `mqs.ini` file to **DiagnosticMessages** in the `qm.ini` file when a queue manager is created.

To display diagnostic messages, use the `mqrc` command.

Attributes of the stanzas



Attention: Service, and a name of a stanza are mandatory.

name=<stanza name>

Name of a stanza. The value must be unique in an ini file.

Service= type of service

This attribute defines a service, where the name of the service is not case sensitive, that is being enabled by this stanza.

For example, to enable syslog as an additional service, enter the following:

```
Service=syslog
```

See “Diagnostic message services” on page 138 and their specific attributes that are available for use with the diagnostic message service stanzas.

You can add the following optional attributes to the stanzas:

- [ExcludeMessage](#)
- [SuppressMessage](#)
- [SuppressInterval](#)
- “Severities” on page 137

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. *msgIds* contain a comma-separated list of message id's from the following:

- 3208 - NativeHA conn couldn't be established
- 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
-  7163 - Job started message (IBM i only)
- 7234 - Number of messages loaded
- 7310 - Report message could not be put
- 7364 - Request to balance appl. sent to QM
-  7366 - Unable to identify QM to handle balance request
- 7367 - Balance request for appl. processed for QM
- 7370 - Streaming queue could not be opened
- 7371 - Put of message to streaming queue failed
- 7490 - Log extent statistics
- 8245 - Entity has insufficient authority to display object
- 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
- 9299 - Channel program started
- 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
- 9917 - Protocol error was ignored due to env var
- 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*. *msgIds* contain a comma-separated list of message identifiers from the following:

- 3208 - NativeHA conn couldn't be established
- 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

 7163 - Job started message (IBM i only)
 7234 - Number of messages loaded
 7310 - Report message could not be put
 7364 - Request to balance appl. sent to QM
 7366 - Unable to identify QM to handle balance request
 7367 - Balance request for appl. processed for QM
 7370 - Streaming queue could not be opened
 7371 - Put of message to streaming queue failed
 7490 - Log extent statistics
 8245 - Entity has insufficient authority to display object
 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
 9299 - Channel program started
 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
 9917 - Protocol error was ignored due to env var
 9999 - Channel program ended abnormally

If the same message ID is specified in both `SuppressMessage` and `ExcludeMessage`, the message is excluded.

This option is not applicable to diagnostic message services defined in `MQ client.ini`.

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 - 86400 seconds. If **SuppressInterval** is not specified, the default value of 30 seconds is used.

Severities

A comma separated list of severity levels, where the name of the severity level is not case sensitive. Allowable values are:

- I (or Information or 0)
- W (or Warning or 10)
- E (or Error or 20 and 30)
- S (or Stop or 40)
- T (or System or 50)

Notes:

1. The default value is `all`
2. Only messages in the selected severity levels are presented to the service.

Alternatively, you can use the plus character (+) which displays the specified error level, and all higher levels. For example, to display all errors:

Severities=E+

Related reference

[“QMErrorLog stanza” on page 133](#)

You use the queue manager error log stanza QMErrorLog in the `qm.ini` file to tailor the operation and contents of IBM MQ error logs.

[“Diagnostic message services” on page 138](#)

The following diagnostic message services and their service specific attributes, specified under the DiagnosticSystemMessages, DiagnosticMessages and DiagnosticMessagesTemplate stanzas of your configuration files can be defined:

Diagnostic message services

The following diagnostic message services and their service specific attributes, specified under the DiagnosticSystemMessages, DiagnosticMessages and DiagnosticMessagesTemplate stanzas of your configuration files can be defined:

The following diagnostic message services are defined:

File

This service sends any unfiltered messages to a file in a similar way to the QMErrorLog service.

Either the existing textual format or the JSON format specified is used depending on the specified

Format. By default, there are three files called AMQERR01.LOG, AMQERR02.LOG, and AMQERR03.LOG or AMQERR01.json, AMQERR02.json, and AMQERR03.json, depending upon the **Format** property, and these rollover based on the configured size.

The following attributes are supported in a File stanza only:

FilePath

The path to where the log files are written. The default is the same location as the AMQERR01.log files, that is system or queue manager. The path must be absolute, but can include replaceable inserts. For example:

+MQ_Q_MGR_DATA_PATH+

The full path to the parent of the queue manager diagnostics messages directory. The default values are:

-   On AIX and Linux platforms: `/var/mqm/qmgrs/<QM_name>`
-  On Windows, `C:\Program Data\IBM\MQ\qmgrs\<QM_name>`

+MQ_DATA_PATH+

The full path to the parent of the system diagnostics messages directory. The default values are:

-   On AIX and Linux platforms: `/var/mqm`
-  On Windows: `C:\Program Data\IBM\MQ`

You must create this path with appropriate permissions, if it is not using the existing errors directory.

FilePrefix

The prefix of the log files. The default is AMQERR.

FileSize

The size at which the log rolls over. The default is 32MB, as with the **ErrorLogSize** property of the [“QMErrorLog stanza” on page 133](#), which is semantically identical.

Note: The **ErrorLogSize** property only applies to the default error log service, not to custom diagnostic services.

You can set the size of the log using the **MQMAXERRORLOGSIZE** environment variable.

Format

The format of the file. The value can be either *text* (for additional QMErrorLog style services) or *json*, which is the default.

The suffix of the file is either *.LOG* or *.json* based on the setting of this attribute.

For example, edit the *qm.ini* file of the queue manager, and add the following stanza:

```
DiagnosticMessages:
  Service = File
  Name = JSONLogs
  Format = json
  FilePrefix = AMQERR
```

After restarting, the queue manager will have *AMQERR0x.json* files in its *ERRORS* directory.

You can define multiple File services. This allows configuration as shown in the following examples, where messages of different tags are split over different sets of logs:

```
DiagnosticMessages:
  Name=ErrorsToFile
  Service=File
  Severities=E+
  FilePrefix=OnlyErrors

DiagnosticMessages:
  Name=NonErrorstoFile
  Service=File
  Severities=I,W
  FilePrefix=InformationandWarning
```

Linux AIX Syslog

The Syslog service is not available on Windows or IBM i

You can define only one Syslog service, and the Syslog service sends any unfiltered messages to syslog using the [JSON format](#) diagnostic messages specification. The information is added to syslog in the order shown in the table, starting with the *msgID* and inserts.

The severity of the message is mapped to the syslog level in the following way:

Table 13. Mapping of message severity to syslog level	
Severity	Level
0	LOG_INFO
10	LOG_WARNING
20	LOG_ERR
30	LOG_ERR
40	LOG_ALERT
50	LOG_ALERT

The following attribute is supported in a syslog stanza only:

Ident

Defines the **ident** value associated with the syslog entries. The default value is *ibm-mq*.

The following example shows error messages being sent to Syslog:

```
DiagnosticMessages:
  Name=ErrorsToSyslog
  Ident=mq
  Service=Syslog
```

See [“Diagnostic message service stanzas” on page 135](#) for more information on generic stanza attributes.

Notes:

1. For the File service only, you can have multiple stanzas, each with a different name. Only the definition, using the final name in the sequence, takes effect.
2. Changes to the value of a stanza come into effect only when the queue manager is restarted.

Multi ExitPath stanza of the qm.ini file

The ExitPath stanza specifies the path for user exit programs on the queue manager system.

Use the ExitPath stanza in the qm.ini file to specify the path for user exit programs on the queue manager system.

Windows **Linux** Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer Exits queue manager properties page.

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

Example stanza

```
ExitPath:
  ExitsDefaultPath=/var/mqm/exits
  ExitsDefaultPath64=/var/mqm/exits64
```

Multi ExitPropertiesLocal stanza of the qm.ini file

The ExitPropertiesLocal stanza specifies information about exit properties on a queue manager.

Use the ExitPropertiesLocal stanza in the qm.ini file, to specify information about exit properties on a queue manager.

Windows **Linux** Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer Cluster queue manager properties page.

Windows Alternatively, on Windows you can specify this information by 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 [“ExitProperties stanza of the mqs.ini file” on page 103](#)). 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.

Use the ExitProperties stanza in the mqs.ini file to specify configuration options used by queue manager exit programs.

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer Extended IBM MQ properties page.

CLWLMode = SAFE (default) |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 (amqzlwao) 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 might 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.

Example stanza

```
ExitPropertiesLocal:
  CLWLMode=SAFE
```

Filesystem stanza of the qm.ini file

Use the Filesystem stanza to configure how the queue manager handles its files.

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.

Example stanza

```
Filesystem:
  ValidateAuth=No
```

JWKS stanza of the qm.ini file

Use the **JWKS** stanza to inform the queue manager how to retrieve a set of public keys which can validate tokens signed by this issuer.

Endpoint=string

Specifies the URL for a JWKS document which provides the public keys required to validate tokens signed by this issuer.

This server specified must provide a secure (HTTPS) connection to retrieve the keys. The queue manager must also have been configured to establish secure HTTP connections, see [HTTPSKeyStore](#).

If unsure of the correct value for this attribute, consult your authentication server administrator.

IssuerName=string

This must be the **IssuerName** which is present in JWT Authentication Tokens supplied/signed by this issuer.

You can establish this by reviewing documentation for your authentication server or examining an example JWT issued by the service.

UserClaim=string

This provides the name of a claim (key/value pair within an authentication token) which the queue manager will use as the IBM MQ user ID when establishing IBM MQ authorities.

If tokens from this endpoint are being used for authentication only, that is **ADOPTCTX** is NO, this attribute is optional, otherwise it is required.

Example stanza

```
JWKS:
  Endpoint=https://myauthserver.example/jwks
  IssuerName=https://myauthserver.example/jwks
  UserClaim=MQUser
```

Related information

[Configuring a queue manager to accept authentication tokens using a JWKS endpoint](#)

Log stanza of the qm.ini file

The Log stanza specifies information about logging on a queue manager.

Use the Log stanza in the qm.ini file to specify information about logging on a queue manager.



Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer **Log queue manager** properties page.

By default, these settings are inherited from the settings that are specified for the default log settings for the queue manager (described in [“LogDefaults stanza of the mqs.ini file” on page 103](#)). Change these settings only if you want to configure this queue manager in a different way.

For more information about calculating log sizes, see [“Calculating the size of the log” on page 733](#).

Note: The limits that are given in the following parameter list are set by IBM MQ. Operating system limits might reduce the maximum possible log size.

LogBufferPages=0 (default) |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.

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 specified number to set the amount of 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.

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 AIX 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 that are specified during queue manager creation cannot be changed for a queue manager.

LogManagement = Manual1 (default)|Automatic | Archive

The method used to manage log extents, either manually or by the queue manager.

You can change the value between Manual1, Automatic and Archive on Linear logging queue managers. The value MUST be set to Automatic on queue managers with REPLICATED logs.

If you change the **LogManagement** value, the change does not take effect until the queue manager is restarted.

If an unrecognized value for the attribute is found, the queue manager will not start until the value is corrected.

 The **LogManagement** property is not valid on IBM i.

Manual1 (default)

You manage the log extents manually. Specifying this option means that the queue manager does not reuse or delete log extents, even when they are no longer required for recovery.

Automatic

Log extents are managed automatically by the queue manager. Specifying this option means that the queue manager is able to reuse or delete log extents as soon as they are no longer required for recovery. No allowance is made for archiving.

Archive

Log extents are managed by the queue manager, but you must notify the queue manager when archiving of each log extent is complete.

Specifying this option means that the queue manager is free to reuse or delete a log extent, as soon as the queue manager has been notified that an extent no longer required for recovery has been archived.

You perform this notification using the **RESET QMGR** MQSC command or [Reset Queue Manager PCF](#) command.

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:

-  **Windows** C:\ProgramData\IBM\MQ\log in Windows.
-  **Linux**  **AIX** /var/mqm/log in AIX 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.



On AIX 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.

LogPrimaryFiles = 3 (default) |2-254 (Windows)|2-510 (AIX 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 AIX 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 AIX 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 (default) |1-253 (Windows)|1-509 (AIX 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 AIX 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 AIX 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.

LogType = CIRCULAR (default) |LINEAR|REPLICATED

The type of logging to be used by the queue manager. The default is CIRCULAR. For more information about creating a queue manager with the type of logging that you require, see the description of the **LogType** attribute in [“LogDefaults stanza of the mq.ini file”](#) on page 103.

CIRCULAR

Start restart recovery by using the log to roll back transactions that were in progress when the system stopped.

See [“Types of logging”](#) on page 727 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 727 for a fuller explanation of linear logging.

REPLICATED

Used by a Native HA group to replicate log data from the active instance to the replica instances. This is equivalent to a linear log with automatic log management and automatic media images enabled.

See [“Types of logging”](#) on page 727 for a fuller explanation of replicated logging.

Note: The **LogType** of a queue manager cannot be changed by modifying this attribute in the `qm.ini` file. To change the **LogType** of a queue manager you must use the **migmqlog** command.

LogWriteIntegrity=SingleWrite|DoubleWrite|TripleWrite (default)

The method the logger uses to reliably write log records.

TripleWrite (default)

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.

For more information, see [“LogWriteIntegrity - using SingleWrite or TripleWrite” on page 145](#).

Example stanza

```
Log:
LogPrimaryFiles=3
LogSecondaryFiles=2
LogFilePages=4096
LogType=CIRCULAR
LogBufferPages=0
LogPath=/var/mqm/log/saturn!queue!manager/
```

Note: The value of zero for **LogBufferPages** gives a value of 512.

LogWriteIntegrity - using SingleWrite or TripleWrite

The setting of the **LogWriteIntegrity** option, in the Log stanza of the `qm.ini` file, determines the algorithm that is used by logger in IBM MQ to write out log records to the recovery log. The default setting is TripleWrite and this setting is both safe in almost every possible scenario.

The setting of **LogWriteIntegrity** has any effect at all, only when a partial log page is to be written. For a queue manager with a reasonable amount of concurrent activity this scenario rarely occurs.

SingleWrite

SingleWrite selects an algorithm which, under very unusual circumstances, can perform better than the default TripleWrite setting. The SingleWrite setting is safe, only if the underlying storage platform can absolutely guarantee under all circumstances that 4KB pages written synchronously to the IBM MQ recovery log are atomically written.

You should use the SingleWrite setting, only if the file-system or 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 should be either the before image, or the after image, and that no intermediate state should be possible. In all other cases you should use TripleWrite.

On a system with sufficient concurrency the queue manager writes only full pages of log data, and if a high percentage of full pages is achieved, there is no significant performance difference between SingleWrite and TripleWrite.

On a system with little concurrency there can be a significant performance advantage to SingleWrite, however the preferred solution is typically to increase concurrency, rather than to use SingleWrite.

Note that it can be difficult to reliably determine the atomicity of 4KB writes, and changes to the underlying software or hardware might invalidate any such guarantee.

If you are in any doubt that your storage infrastructure makes the required guarantees now, and at any time in the future in all circumstances, you should use TripleWrite.

LU62 stanza of the qm.ini file (Windows only)

The LU62 stanza specifies SNA LU 6.2 protocol configuration parameters. These parameters override the default attributes for channels.

Use the LU62 stanza in the `qm.ini` file to specify SNA LU 6.2 protocol configuration parameters. They override the default attributes for channels.

Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer LU6.2 queue manager properties page.

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.

NativeHAInstance stanza of the qm.ini file

The NativeHAInstance stanza specifies how the three nodes in a Native HA configuration can communicate with each other.

Note: The NativeHAInstance stanza is required for Native HA and Native HA CRR configurations. It is not required for Native HA IRR configurations, and supplying it causes the creation of a Native HA IRR configuration to fail.

There are three NativeHAInstance stanzas, one for each queue manager instance in the Native HA group (including the local instance). They have the following attributes:

Name

Specify the instance name that you used when you created the queue manager instance.

ReplicationAddress

Specify the hostname, IPv4 dotted decimal or IPv6 hexadecimal format address of the instance. You can specify the address as a hostname, IPv4 dotted decimal, or IPv6 hexadecimal format address. The replication address must be resolvable and routable from each instance in the group. The port number to use for the log replication must be specified in brackets, for example:

```
ReplicationAddress=host1.example.com(4444)
```

Example stanza

The following example shows the NativeHAInstance stanza used in the `qm.ini` file to specify the three nodes of a native HA configuration.

```
NativeHAInstance:
  Name=node-1
  ReplicationAddress=host1.example.com(4444)
NativeHAInstance:
  Name=node-2
  ReplicationAddress=host2.example.com(4444)
NativeHAInstance:
  Name=node-3
  ReplicationAddress=host3.example.com(4444)
```

Related concepts

[“NativeHALocalInstance stanza of the qm.ini file” on page 147](#)

The NativeHALocalInstance stanza controls operation of a Native HA configuration.

NativeHALocalInstance stanza of the qm.ini file

The NativeHALocalInstance stanza controls operation of a Native HA configuration.

The NativeHALocalInstance stanza is added to the qm.ini file on each of the nodes in the Native HA configuration.

Name

The name of the NativeHALocalInstance stanza, taken from the log replica instance name specified when the Native HA queue manager is created.

You can optionally add the following attributes to the NativeHALocalInstance stanza:

KeyRepository

The full path and the file name of the key repository that holds the digital certificate that is used to protect log replication traffic. If the file extension is not specified, it is assumed to be .kdb.

If the KeyRepository stanza attribute is omitted, log replication data is exchanged between instances in plain text.

KeyRepositoryPassword

The key repository is secured with a password as it contains sensitive information. To be able to access the key repository contents, IBM MQ must be able to retrieve the key repository password. If the password is not stored in a key repository stash file, you can supply the password in the KeyRepositoryPassword attribute. For example:

```
KeyRepositoryPassword=passw0rd
```



Attention: If you supply the password by using this attribute, encrypt the password with the IBM MQ password protection system. For more information, see [“Encrypting the key repository password” on page 149](#).

InitialKeyFile

Specify this attribute if the key repository password that is specified with the KeyRepositoryPassword attribute is encrypted with a specific initial key. The name of the file that contains the initial key can be specified by using the **-sf** parameter when the **runmqicred** command is used to encrypt the key repository password.

Set the value of this attribute to the name of the file that contains the initial key used to encrypt the password. For example, if a file called mykey.key contains the initial key:

```
InitialKeyFile=/mykey.key
```

For more information, see [“Encrypting the key repository password” on page 149](#).

CertificateLabel

The certificate label identifying the digital certificate to use for protection of log replication traffic. If KeyRepository is provided but CertificateLabel is omitted, a default value of *ibmwebsphermqueue_manager* is used.

CipherSpec

The CipherSpec to use to protect log replication traffic. If this stanza attribute is provided, KeyRepository must also be provided. If KeyRepository is provided but CipherSpec is omitted, a default value of ANY is used.

LocalAddress

The local network interface address that accepts log replication traffic. If this stanza attribute is provided it identifies the local network interface and/or port using the format “[addr]([port])”. The network address can be specified as a hostname, IPv4 dotted decimal, or IPv6 hexadecimal format. If this attribute is omitted, the queue manager attempts to bind to all network interfaces, it uses the port specified in the ReplicationAddress in the NativeHAInstances stanza matching the local instance name.

HeartbeatInterval

The heartbeat interval defines how often in milliseconds an active instance of a Native HA queue manager sends a network heartbeat. The valid range of the heartbeat interval value is 500 (0.5 seconds) to 60000 (1 minute), a value outside of this range causes the queue manager to fail to start. If this attribute is omitted, a default value of 5000 (5 seconds) is used. Each instance must use the same heartbeat interval.

HeartbeatTimeout

The heartbeat timeout defines how long a replica instance of a Native HA queue manager waits before it decides that the active instance is unresponsive. The valid range of the heartbeat interval timeout value is 500 (0.5 seconds) to 120000 (2 minutes). The value of the heartbeat timeout must be greater than or equal to the heartbeat interval.

An invalid value causes the queue manager to fail to start. If this attribute is omitted a replica waits for 2 x HeartbeatInterval before starting the process to elect a new active instance. Each instance must use the same heartbeat timeout.

RetryInterval

The retry interval defines how often in milliseconds a Native HA queue manager should retry a failed replication link. The valid range of the retry interval is 500 (0.5 seconds) to 120000 (2 minutes). If this attribute is omitted a replica waits for 2 x HeartbeatInterval before retrying a failed replication link.

SSLFipsRequired

Specifies whether only FIPS certified algorithms are used if cryptography is used in sending log replication traffic. Set to Yes or No.

EncryptionPolicySuiteB

Specifies whether log replication traffic uses Suite-B compliant cryptography and what level of strength is used. Set to one of the following 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.

CompressionThreshold

Sets a byte threshold that, when crossed, triggers compression of log data. Log data larger than the threshold value is compressed. A value of 0 (the assumed default) turns off all compression, a value of 1 compresses every log append. The maximum value is 268435456 (256 MB).

V 10.00 V 10.00 For Native HA IRR configurations (that require low network latency for synchronous replication), it is assumed that it is more efficient not to use compression. LZ4 fast compression is only applied to synchronous log appends when compression is enabled and the size of the log data being sent exceeds the threshold of both GroupCompressionThreshold and CompressionThreshold. If either value is 0, the synchronous log append traffic is sent uncompressed.

LZ4Acceleration

A tuning parameter that controls how the LZ4 algorithm looks for compressible sequences in the log data. Every time the acceleration value increases by 1, the algorithm does not look quite as carefully for a compressible sequence, exchanging compression ratio for a small performance gain. The minimum (and assumed default) value is 1, the maximum is 65537.

V 10.00 **GroupName**

Specifies the group name that this instance is a member of. This is required if:

- If a NativeHARecoveryGroup stanza is specified (regardless of whether it is enabled or not).
- If the GroupRole is set to Recovery.

GroupRole

Specifies the current role of the group that the instance belongs to. Set to Live or Recovery.

GroupLocalAddress

Specifies the replication network interface and port to advertise for the group. If not specified, defaults to all network interfaces and port 9415.

GroupCipherSpec

Optionally overrides the CipherSpec when establishing a TLS connection between groups. The group connection must be secured using TLS.

GroupCertificateLabel

Optionally overrides the certificate label to use for mutual authentication when establishing a TLS connection between groups.

GroupCompressionThreshold

Sets a byte threshold that, when crossed, triggers compression of log data when sent to a recovery group. Log data larger than the threshold value is compressed. A value of 1 (the assumed default) compresses every log append. A value of 0 turns off all compression of log data being sent to a recovery group. The maximum value is 268435456 (256 MB).

Encrypting the key repository password

The key repository password can be protected by using either the IBM MQ password protection system, or a key repository stash file. For more information about these two methods, see [Encrypting key repository passwords](#).

If the repository password is specified by using the KeyRepositoryPassword attribute in the NativeHALocalInstance stanza, encrypt the password by using the password protection system. Use the IBM MQ **runmqicred** command to encrypt the password. The command returns the encrypted password that can be specified in the KeyRepositoryPassword attribute.

Use a unique initial key to encrypt the password securely. The name of the file that contains the initial key can be specified by using the **-sf** parameter to the **runmqicred** command. If you do not supply a unique key, the default key is used.

If you encrypt the key repository password with a unique initial key, you must also supply the same initial key by using the InitialKeyFile attribute in the NativeHALocalInstance stanza.

Example stanza

The following example shows the NativeHALocalInstance stanza used in the `qm.ini` file to specify the local name of a node.

```
NativeHALocalInstance:
  Name=node-1
```

The following example shows the NativeHALocalInstance stanza used in the `qm.ini` file to specify the local name of a node, and to specify the group details because the instance is part of a Native HA cross-region replication configuration

```
NativeHALocalInstance:
  Name=alpha-1
  GroupName=alpha
  KeyRepository=/var/mqm/qmgrs/MYQM/ssl/key
  CertificateLabel=MyCertificate
  CipherSpec=NULL
  GroupRole=Live
  GroupCipherSpec=ANY_TLS12
  GroupLocalAddress=(4445)
```

Related concepts

[“NativeHAInstance stanza of the qm.ini file” on page 146](#)

The NativeHAInstance stanza specifies how the three nodes in a Native HA configuration can communicate with each other.

Related reference

[runmqicred \(protect IBM MQ Client passwords\)](#)

NativeHARecoveryGroup stanza of the qm.ini file

The NativeHARecoveryGroup stanza controls operation of a Native HA cross-region replication (CRR) or Native HA in-region replication (IRR) configuration.

The NativeHARecoveryGroup stanza is added to the `qm.ini` file on each of the nodes when you create a Native HA CRR or IRR configuration.

GroupName

The name of the **other** group forming the Native HA CRR or IRR configuration. The group name of **this** instance is specified in the NativeHALocalInstance stanza.

Enabled

Specifies whether the Native HA recovery group is enabled, if not specified, defaults to Yes. When the Enabled attribute is set to No, the replication configuration is ignored. This setting can be used for an unplanned failover to force through a requested role change when communication with the other group is not possible.

ReplicationAddress

Specifies a hostname and port that addresses the replication traffic port on the other group in the Native HA CRR or IRR configuration. The address can be a single service address and port or a comma-separated list.

ShortRetryTimer

Specifies the interval to wait between attempts to connect to a Recovery group in the short retry phase. Specified in milliseconds, defaults to 2500 (2.5 seconds).

ShortRetryCount

Specifies the number of attempts to make a connection to the Recovery group during the short retry phase.

LongRetryTimer

Specifies the interval to wait between attempts to connect to a Recovery group in the long retry phase. The long retry phase continues indefinitely until connection is made, or until the configuration is changed to comment out or remove the ReplicationAddress. Specified in milliseconds, defaults to 60000 (60 seconds).

RetryReporting

Set to Phase or All to specify whether each retry attempt is reported, or just the phase (long or short). If not specified, defaults to Phase.

StatusInterval

Specifies the longest time interval that the Recovery group will wait before sending a Recovery log status. Specified in milliseconds, defaults to 10000 (10 seconds).

StatusCommitThreshold

Specifies the maximum number of bytes that the Recovery group commits before sending a Recovery log status. Specified in bytes and defaults to 0, which means that StatusInterval alone is used to determine when Recovery log status is sent.

SyncReplication

Specifies whether a Native HA configuration uses synchronous (Yes) or asynchronous (No) replication to communicate with the recovery group. The default is No. You can only set this option to Yes if you are specifying a Native HA IRR configuration. The Native HA CRR configuration always uses asynchronous replication.

SyncConsistency

If SyncReplication is set to Yes (which is only available for the Native HA IRR configurations) you can use SyncConsistency to specify behavior if it is not synchronized (either because the live instance cannot connect to the recovery instance or because replication has fallen behind). Set to Strict to specify that data consistency is prioritized, set to Eventual to specify that message or service availability is prioritized. The default is Strict, which indicates that both Live and Recovery groups must be connected and in-sync before a queue manager can start and progress new log writes. If set to Eventual, the Live group makes a single attempt to connect to the Recovery group and synchronize. If the Live group is unable to establish a replication link, it starts to progress log writes without waiting for the acknowledgement from the Recovery group.

Example stanza

The following is an example NativeHARecoveryGroup stanza:

```
NativeHARecoveryGroup:
  GroupName=alpha
  ReplicationAddress=alpha-cluster.fyre.ibm.com(4445)
```

Related concepts

[“NativeHALocalInstance stanza of the qm.ini file” on page 147](#)

The NativeHALocalInstance stanza controls operation of a Native HA configuration.

[“NativeHAInstance stanza of the qm.ini file” on page 146](#)

The NativeHAInstance stanza specifies how the three nodes in a Native HA configuration can communicate with each other.

NETBIOS stanza of the qm.ini file (Windows only)

The NETBIOS stanza in the qm.ini file specifies NetBIOS protocol configuration parameters. These parameters override the default attributes for channels.

Use the NETBIOS stanza in the qm.ini file, to specify NetBIOS protocol configuration parameters. They override the default attributes for channels.

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer Netbios queue manager properties page.

LocalName= *name*

The name by which this machine is known on the LAN.

AdapterNum = 0 (default) | *adapter_number*

The number of the LAN adapter. The default is adapter 0.

NumSess = 1 (default) | *number_of_sessions*

The number of sessions to allocate. The default is 1.

NumCmds = 1 (default) | *number_of_commands*

The number of commands to allocate. The default is 1.

NumNames = 1 (default) | *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.

Related concepts

[“Defining the IBM MQ local NetBIOS name” on page 279](#)

The local NetBIOS name used by IBM MQ channel processes can be specified in three ways.

RestrictedMode stanza of the qm.ini file

The RestrictedMode stanza specifies the name of the group that contains members that are allowed to run MQI applications, update all IPCC resources and change the contents of some queue manager directories. This stanza applies to AIX and Linux systems only.

The RestrictedMode stanza is set by the **-g** option on the **crtmqm** command. 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 MQCONNEX call issued by a user who is not in the application group fails with reason code **MQRC_Q_MGR_NOT_AVAILABLE**.

Important: On many operating systems, in order for the addition of a user to a group to be recognized, the user in question must log off and log back on.

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.

 Further information about configuring the IBM MQ authorization service can be found in [Setting up security on AIX, Linux, and Windows systems](#).

Only use IBM MQ restricted mode when the control provided by the authorization service does not provide sufficient isolation of queue manager resources.

Related reference

crtmqm (create queue manager)

Multi Security stanza of the qm.ini file

The Security stanza specifies 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.

Note: Setting `ClusterQueueAccessControl=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 `ClusterQueueAccessControl=Xmitq` or upgrade the custom Authorization Service to `MQZAS_VERSION_6` or greater.

Windows 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.

V1000 DisablePolicyFIPSCheck=YES|NO

Allows Advanced Message Security (AMS) policies to be defined with algorithms that do not meet FIPS compliance if the queue manager is enabled for FIPS.

Example stanza

```
Security:
  ClusterQueueAccessControl=Xmitq
  GroupModel=GlobalGroups
```

Multi Service stanza of the qm.ini file

The Service stanza is used to make changes to installable services. This stanza contains the name of the service and the number of entry points defined for the service.

Note: Windows Linux There are significant implications to changing installable services and their components. For this reason, the installable services are read-only in IBM MQ Explorer.

For each component within a service, you must also specify the name and path of the module containing the code for that component. Use the [ServiceComponent](#) stanza for this.

The **Service** and **ServiceComponent** stanzas can occur in any order and the stanza keys under them can also occur in any order. For either of these stanzas, all the stanza keys must be present. If a stanza key is duplicated, the last one is used.

At start up, the queue manager processes each service component entry in the configuration file in turn. It then loads the specified component module, invoking the entry point of the component (which must be the entry point for initialization of the component), passing it a configuration handle.

Name = AuthorizationService (default) |NameService

The name of the required service.

AuthorizationService

For IBM MQ, the **AuthorizationService** component is known as the object authority manager, or OAM. The **Service** stanza and its associated **ServiceComponent** stanza are added automatically when the queue manager is created, but can be overridden by the MQSNOAUT environment variable. Add other **ServiceComponent** stanzas manually.

► Linux ► AIX The following examples of stanzas in the `qm.ini` file define two authorization service components on IBM MQ for AIX. `MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.

```
Service:
  Name=AuthorizationService
  EntryPoints=13

ServiceComponent:
  Service=AuthorizationService
  Name=MQSeries.UNIX.auth.service
  Module=MQ_INSTALLATION_PATH/lib/amqzfu
  ComponentDataSize=0

ServiceComponent:
  Service=AuthorizationService
  Name=user.defined.authorization.service
  Module=/usr/bin/udas01
  ComponentDataSize=96
```

► Linux ► AIX The ServiceComponent stanza `MQSeries.UNIX.auth.service` defines the default authorization service component, the OAM. If you remove this stanza and restart the queue manager, the OAM is disabled and no authorization checks are made.



CAUTION: ► Windows The Windows Registry contains critical information, including file locations, drivers, and start-up processes. IBM MQ relies on these specific registry settings to function correctly. Changing these settings without proper information can cause the IBM MQ application to become unusable or fail completely as incorrect changes can disrupt its essential functions.

► Windows You can also add the **SecurityPolicy** attribute using the IBM MQ services. The **SecurityPolicy** attribute applies only if the service specified on the Service stanza is the authorization service, that is, the default OAM. The **SecurityPolicy** attribute allows you to specify the security policy for each queue manager. The possible values are:

Default

Specify `Default` if you want 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

Requires that an NT SID is passed to the OAM when performing security checks.

► Windows The ServiceComponent stanza `MQSeries.WindowsNT.auth.service` defines the default authorization service component, the OAM. If you remove this stanza and restart the queue manager, the OAM is disabled and no authorization checks are made.

NameService

No name service is provided by default. If you require a name service, you must add the **NameService** stanza manually.

► Linux ► AIX The following examples of AIX and Linux `qm.ini` file stanzas for the name service specify a name service component provided by the (fictitious) ABC company.

```
# Stanza for name service
Service:
  Name=NameService
  EntryPoints=5

# Stanza for name service component, provided by ABC
ServiceComponent:
  Service=NameService
  Name=ABC.Name.Service
  Module=/usr/lib/abcname
  ComponentDataSize=1024
```

Note: ► Windows On Windows systems, NameService stanza information is stored in the Registry.

EntryPoints= *number-of-entries*

The number of entry points defined for the service.

This includes the initialization and termination entry points.

► Windows **SecurityPolicy= Default|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:

Default

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.

For more information, see [Windows security identifiers \(SIDs\)](#).

► Linux ► AIX **SecurityPolicy=user|group|UserExternal|default**

On AIX and Linux systems, the value specifies whether the queue manager uses user-based or group-based authorization. Values are not case sensitive.

The value can be one of the following values:

group

The queue manager uses group-based authorization. Authority to access a resource is granted to a group.

A user receives the aggregate of all the authorities that are granted to each group that it belongs to.

User IDs and groups must be defined to the local operating system.

user

The queue manager uses user-based authorization. Authority to access a resource can be granted to a group, or a specific user ID.

A user receives the aggregate of the following authorities:

- Authorities that are granted to the specific user.
- Authorities that are granted to each group that the user belongs to.

User IDs and groups must be defined to the local operating system.

UserExternal

The queue manager uses user-based authorization. However, authorities can be granted to user IDs that are not known to the local operating system.

Authority to access a resource can be granted to a group, or a specific user ID.

A user receives the aggregate of the following authorities:

- Authorities that are granted to the specific user.
- Authorities that are granted to each group that the user belongs to.

If a user is not known to the local operating system, it is considered to belong to only the nobody group. For more information about groups, see [Principals and groups on AIX, Linux, and Windows](#). The user ID must be no more than 12 characters long, and must conform to the [Rules for naming IBM MQ objects](#).

You can modify existing queue managers to use this additional option without losing any current configuration.

This is the default value if the AuthToken stanza is specified.

default

The queue manager uses group-based authorization. The behavior is the same as for the group option.

This is the default value if the AuthToken stanza is not specified.

Restart the queue manager for changes to the attribute value to become effective.

Note:   From IBM MQ 9.4.0, if the AuthToken stanza is specified, the effective value of the **SecurityPolicy** attribute of the Service stanza is set to UserExternal. Token authentication is not available if **SecurityPolicy** is explicitly set to Group in the Service stanza. If **SecurityPolicy** is set to Group, remove the **SecurityPolicy** attribute from the Service stanza, then restart the queue manager. For more information, see [“AuthToken stanza of the qm.ini file” on page 122](#).

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 more 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 more 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 more 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.

AllowLongUID=YES | NO (default)

The **AllowLongUID** attribute applies only if the service specified is the default authorization service, that is, the OAM, and IDPWOS authentication is being used.

When AllowLongUID=YES is specified, OAM authorities can be defined against user identities of up to 1024 characters, and are applied when applications connect and supply authenticated credentials for those identities. In certain places throughout the queue manager, notably the MQMD **userId** field, this identity is truncated to a form of the first 11 characters suffixed with a '+' character to indicate that truncation has taken place (see also [User identity length limitations \(MQMD and downstream authorization\)](#)). Before you enable this mode of operation, consider whether this truncation could affect overall application security.

For more information about installable services and components, see [Installable services and components for AIX, Linux, and Windows](#).

For more information about security services in general, see [Setting up security on AIX and Linux systems](#).

Example stanza

```
Service:
  Name=AuthorizationService
  EntryPoints=14
```

Related concepts

[Installable services and components for AIX, Linux, and Windows](#)

Related reference

[Installable services and components on IBM i](#)

[Installable services reference information](#)

Multi ServiceComponent stanza of the qm.ini file

The ServiceComponent stanza specifies information for the service component. You must specify service component information when you add a new installable service. The authorization service stanza is present by default, and the associated component, the OAM, is active.

The **Service** and **ServiceComponent** stanzas can occur in any order and the stanza keys under them can also occur in any order. For either of these stanzas, all the stanza keys must be present. If a stanza key is duplicated, the last one is used.

At startup time, the queue manager processes each service component entry in the configuration file in turn. It then loads the specified component module, invoking the entry point of the component (which must be the entry point for initialization of the component), passing it a configuration handle.

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.

Example stanza

```
ServiceComponent:
  Service=AuthorizationService
  Name=MQSeries.UNIX.auth.service
  Module=amqzfu
  ComponentDataSize=0
```



CAUTION: **Windows** The Windows Registry contains critical information, including file locations, drivers, and start-up processes. IBM MQ relies on these specific registry settings to function correctly. Changing these settings without proper information can cause the IBM MQ application to become unusable or fail completely as incorrect changes can disrupt its essential functions.

For further examples showing an AuthorizationService stanza and its associated ServiceComponent stanzas and a NameService stanza and its associated ServiceComponent stanza, see [“Service stanza of the qm.ini file” on page 153](#).

Related concepts

[Installable services and components for AIX, Linux, and Windows](#)

Related reference

[“Service stanza of the qm.ini file” on page 153](#)

The Service stanza is used to make changes to installable services. This stanza contains the name of the service and the number of entry points defined for the service.

[Installable services and components on IBM i](#)

[Installable services reference information](#)

SPX stanza of the qm.ini file (Windows only)

The SPX stanza specifies SPX protocol configuration parameters. These parameters override the default attributes for channels.

Use the SPX stanza in the `qm.ini` file, to specify SPX protocol configuration parameters.

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer SPX queue manager properties page.

Socket = 5E86 (default) | *socket_number*

The SPX socket number in hexadecimal notation. The default is X'5E86'.

BoardNum = 0 (default) | *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 159](#).

<i>Table 14. Default outstanding connection requests (SPX)</i>	
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 159](#), 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.

SSL stanza of the qm.ini file

The SSL stanza is used to configure the TLS channels on a 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.

Attributes of the SSL stanza

Use the SSL stanza in the `qm.ini` file to configure how 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.

AllowedQuantumSafeAlgorithms=ALL (default) | NONE | HYBRIDMLKEM | MLKEM

Specifies which quantum safe algorithms the queue manager supports as a TLS server or TLS client. This can be a comma separated list of algorithms or a single special value of ALL or NONE. Duplicates or invalid values will result in an error and a default setting. The default is ALL.

- ALL: Enables all quantum safe algorithms.
- NONE: Disables all quantum safe algorithms.
- HYBRIDMLKEM: Enables the Hybrid ML-KEM algorithm.
- MLKEM: Enables the ML-KEM algorithm.

OutboundSNI = CHANNEL | HOSTNAME

If **OutboundSNI** is set to CHANNEL, SNI capable clients set SNI to the target IBM MQ channel name to the remote system when initiating a TLS connection.

If this attribute is set to HOSTNAME, SNI capable clients set the SNI header to the hostname, causing outbound connection requests to receive the default certificate of the remote queue manager during the TLS handshake, and so per-channel certificates cannot be used.

Note: If **OutboundSNI=HOSTNAME** is used to connect to a remote channel with a certificate label configured, the connection is rejected with an MQRC_SSL_INITIALIZATION_ERROR and an AMQ9673 message is printed in the remote queue manager error logs.

AllowOutboundSNI = YES (default) | NO

If enabled, SNI capable clients will set SNI to the target IBM MQ channel name to the remote system when initiating a TLS connection. If this attribute is set to NO, SNI capable clients will not set the SNI header causing outbound connection requests to receive the remote queue manager's default certificate during the TLS handshake, and so per-channel certificates cannot be used.



Attention:  From IBM MQ 9.3.0 the **AllowOutboundSNI** property is deprecated, and is available for backwards-compatibility purposes only.

AllowOutboundSNI set to YES provides the same function as **OutboundSNI** set to CHANNEL, while **AllowOutboundSNI** set to NO provides the same function as **OutboundSNI** set to HOSTNAME.

If both **AllowOutboundSNI** and **OutboundSNI** attributes are present in the SSL stanza, the setting of **OutboundSNI** takes precedence.

AllowedCipherSpecs=name|name list|ALL

Specifies a custom list of CipherSpecs that are ordered and enabled for use with IBM MQ channels on Multiplatforms. Specifying **AllowedCipherSpecs** restricts IBM MQ listeners from accepting incoming channel start requests, unless they use one of the named CipherSpecs.

- A single CipherSpec name.
- A comma separated list of IBM MQ CipherSpec names to re-enable.
- ALL The special value of ALL, representing all CipherSpecs (not recommended).

Note: You should not select **ALL** CipherSpecs, as this enables all weak and insecure cryptographic algorithms.

For more information, see [Providing a custom list of ordered and enabled CipherSpecs on IBM MQ for Multiplatforms in CipherSpec order in TLS handshake.](#)

AllowTLSV13=Y|YES|T|TRUE (default)|N|NO|F|FALSE

Specifies whether a queue manager is able to use the TLS 1.3 CipherSpecs.

- Y (default), YES (default), T (default), or TRUE (default): Enables TLS 1.3 which allows the queue manager to use the TLS 1.3 CipherSpecs.
- N, NO, F, or FALSE: Disables TLS 1.3, which means that the queue manager is unable to use the TLS 1.3 CipherSpecs.

For more information, see [Enabling CipherSpecs.](#)

CDPCheckExtensions=YES|NO (default)

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

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

HTTPSCertRevocation=REQUIRED(default)|OPTIONAL|DISABLED

Sets the level of CRL/OCSP extension checks for HTTPS certificates to a JWKS endpoint.

- REQUIRED (default): Check the certificate's revocation status and make sure that it is resolved.
- OPTIONAL: Check the certificate's revocation status but tolerate if it is unknown.
- DISABLED: Do not check the certificate's revocation status.

HTTPSCertValidation=HOSTNAMECN(default)|ANY|NONE

Sets the level of certificate peer name validation performed when establishing HTTPS connections to a JWKS endpoint.

- HOSTNAMECN (default): Check that the peer's certificate is valid, signed and belongs to the remote host.
- ANY: Check that the peer's certificate is valid and signed, but do not fail if it does not belong to the remote host (if the CN field does not match remote host's name).
- NONE: Do not validate the server's certificate.

HTTPSKeyStore=string

The string provides the path to a pkcs12 key repository which can be used by the queue manager as a trust store when creating outgoing HTTPS connections, for example to a JWKS endpoint.

This file must be encrypted and accompanied by a ‘stash’ file of the same name , that is a file with a .sth extension, used when the queue manager needs to access this file. By default, if this attribute is not specified, the queue manager looks in the ssl subdirectory of the queue manager data for a file named mqdefcer.p12.

If this attribute is updated, you must restart the queue manager to begin using the new key repository file specified.

See [Creating a key repository for use as a TLS truststore](#) for guidance on creating a truststore based off your operating system environment.

ALW **MinimumRSAKeySize=int**

Specifies the minimum keysize that RSA certificates must have in order to be accepted during a TLS handshake. Allows any value equal to 0 or above. Defaults to **LTS** 1 or V 10.0.0 2048 if not specified.

OCSPAuthentication=REQUIRED (default) |WARN|OPTIONAL

Specifies the action to be taken when a revocation status cannot be determined from an OCSP server. If OCSP checking is enabled, a 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 (default):** 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 (default) |NO

Specifies whether TLS channels on this queue manager try to check OCSP servers that are named in AuthorityInfoAccess certificate extensions.

- **YES (default) :** TLS channels try to check OCSP servers to determine whether a digital certificate is revoked. This value is the default.
- **NO:** TLS channels do not try to check OCSP servers.

ALW **OCSPTimeout= number**

The number of seconds to wait for an OCSP responder when performing a revocation check.

From IBM MQ 9.3.0, if a value of 0 is set, the default timeout of 30 seconds is used.

If no value is set, then the IBM MQ default of 30 seconds is used.

ALW **PeerCertChainValidation=string**

The string can be one of two values:

- **Usepeerchain [Default]:** The certificate chain provided by the peer can be used to bridge any trust chain gaps while validating certificates. With the exception of the root certificate.
- **Truststoreonly [Not recommended]:** Only certificates in the trust store will be used for validating the peer's certificate.

V 10.0.0 **ALW** **RFC5280EKUChecks = TRUE (default) | FALSE**

Validates the Extended Key Usage (EKU) extension of the remote queue manager certificate where the local queue manager is acting as client.

This attribute has the following possible values:

TRUE

If the remote queue manager certificate has specified an EKU that does not include `serverAuth`, then the connection is rejected. This is the default value.

FALSE

The EKU not validated whether the remote queue manager certificate has specified an EKU or not.

SSLHTTPConnectTimeout= *number*|0

The number of seconds to wait for a network connection to be successfully established to a HTTP server when performing a revocation check.

If no value is set, then the IBM MQ default of 0 (off) is used.

SSLHTTPProxyName= *string*

The string is either the host name or network address of the HTTP Proxy server that is to be used by IBM Global Security Kit (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 32-bit clients on AIX, the network address can only be an IPv4 address.

On other platforms, the network address 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.

Example stanza

```
SSL:
    OutboundSNI=CHANNEL
    AllowedCipherSpecs=TLS13 CipherSpec list
    AllowTLSV13=Y
    CDPCheckExtensions=NO
    MinimumRSAKeySize=1
    OCSPAuthentication=REQUIRED
    OCSPCheckExtensions=YES
    OCSPTimeout=30
    PeerCertChainValidation=Usepeerchain
    SSLHTTPConnectTimeout=0
```

Notes:

- The default value for **OutboundSNI** is **Channel1**.
- The **TLS13 CipherSpec** list is a list of specific CipherSpecs not the alias ciphers. If you require only TLS1.3 ciphers, you must list them. For example:
TLS_CHACHA20_POLY1305_SHA256
TLS_AES_256_GCM_SHA384
TLS_AES_128_GCM_SHA256
TLS_AES_128_CCM_SHA256
TLS_AES_128_CCM_8_SHA256
- The default value for **AllowTLSV13** is Y unless you have enabled weak ciphers, in which case it turns off (unless you explicitly turn it on.)
- The values for **CDPCheckExtensions** can be only Yes or No.
- The values for **PeerCertChainValidation** can be only Usepeerchain or Truststoreonly.

Multi Subpool stanza of the qm.ini file

This stanza is created by IBM MQ. Do not change it.

The Subpool stanza, 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.

Multi TCP stanza of the qm.ini file

The TCP stanza specifies Transmission Control Protocol/Internet Protocol (TCP/IP) configuration parameters. These parameters override the default attributes for channels.

Use the TCP stanza in the `qm.ini` file, to specify TCP/IP configuration parameters.

Windows Linux Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer SPX TCP queue manager properties page.

Port = 1414 (default) | port_number

The default port number, in decimal notation, for TCP/IP sessions. The *well known* port number for IBM MQ is 1414.

Windows Library1= DLLName1 (Windows only)

The name of the TCP/IP sockets DLL.

The default is WSOCK32.

Multi SecureCommsOnly = NO|N|FALSE|F (default) |TRUE|T|YES|Y

Specify whether plain text communication is allowed or not.

SecureCommsOnly=NO|N|FALSE|F

Plain text communication is allowed, and a warning message is output when the queue manager starts.

SecureCommsOnly=YES|Y|TRUE|T

Plain text communication is not allowed, and an information message is output when the queue manager starts.

KeepAlive = NO (default) |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 15 on page 164](#).

Table 15. Default outstanding connection requests (TCP)	
Platform	Default ListenerBacklog value
Windows Windows Server	100
Linux Linux	100
AIX 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 15 on page 164](#), 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 (default) |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 (default)

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.

 New queue managers are automatically created with a default setting of 0 (see [“Example stanza” on page 166](#)).

RcvBuffSize=number|0 (default)

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.

 New queue managers are automatically created with a default setting of 0 (see [“Example stanza” on page 166](#)).

RcvSndBuffSize=number|0 (default)

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.

 New queue managers are automatically created with a default setting of 0 (see [“Example stanza” on page 166](#)).

RcvRcvBuffSize=number|0 (default)

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.

 New queue managers are automatically created with a default setting of 0 (see [“Example stanza” on page 166](#)).

SvrSndBuffSize=number|0 (default)

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.

Multi New queue managers are automatically created with a default setting of 0 (see “Example stanza” on page 166).

SvrRcvBuffSize=number|0 (default)

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.

Multi New queue managers are automatically created with a default setting of 0 (see “Example stanza” on page 166).

V 9.4.3 V 9.4.0.10 DNSResolutionOrder=list|IPV6, IPV4, ANY(default)

The order in which IBM MQ looks up host names in mixed IPv6 and IPv4 environments. If the **AMQ_NO_IPV6** environment variable is set, IBM MQ acts as if **DNSResolutionOrder=IPV4** is set, no matter what value is configured.

The value of **DNSResolutionOrder** is a comma-separated list with the values IPV6, IPV4, and ANY in the order preferred. These values correspond to IPv6, IPv4 and unspecified address family DNS lookups, where an unspecified lookup can resolve both IPv6 and IPv4 addresses. Each value can appear in the list at most once. If IPV4 or IPV6 are missing from the list, IBM MQ implicitly appends ANY to make sure it can resolve all address types.

Setting the list value to just ANY might give faster performance. IPV6 and IPV4 can be listed in the order reflecting their importance on the local network.

Example stanza

```
TCP:
  SndBuffSize=0
  RcvBuffSize=0
  RcvSndBuffSize=0
  RcvRcvBuffSize=0
  ClntSndBuffSize=0
  ClntRcvBuffSize=0
  SvrSndBuffSize=0
  SvrRcvBuffSize=0
```

Note: **Multi** For new queue managers on Multiplatforms, 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. As shown in the preceding example, new queue managers are automatically created with a default setting of 0 for the send and receive buffers. This applies to new queue managers only. The TCP send and receive buffer settings for queue managers that are migrated from earlier versions of IBM MQ are 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.

Multi TuningParameters stanza of the qm.ini file

The TuningParameters stanza specifies options for tuning the queue manager.

SuppressDspAuthFail=YES|NO (default)

When set to YES, the queue manager suppresses generation of authorization events and the writing of AMQ8077 error messages to the error log when an authorization check fails, if the connection lacks +dsp authority to an object.

ImplSyncOpenOutput=value

ImplSyncOpenOutput is the minimum number of applications that have the queue open for put, before an implicit syncpoint might be enabled for a persistent put, outside of syncpoint. The default value of **ImplSyncOpenOutput** is 2.

This has the effect that if there is only one application that has that queue open for a put operation, **ImplSyncOpenOutput** is switched off.

Specifying **ImplSyncOpenOutput=1** means that an implicit syncpoint is always considered. You can set any positive integer value. If you never want an implicit syncpoint to be added, set **ImplSyncOpenOutput=OFF**.

UniformClusterName=name of cluster

The name of the IBM MQ cluster that you are using as a uniform cluster.

OAMLdapConnectTimeout=time|0 (default)

The maximum time, in seconds, that the LDAP client will wait to establish a TCP connection to the server. If you are supplying multiple LDAP servers through a connection namelist, the timeout applies to each individual connection attempt, and so a connection to the next entry in the namelist is attempted if this timeout is reached.

time has a maximum value of 3600 seconds, and a value of 0, which is the minimum value as well as the default value, means that the wait is unlimited.

OAMLdapQueryTimeLimit=time|0 (default)

The maximum time, in seconds, that the LDAP client will wait to receive a response to an LDAP request from the server, once a connection has been established, and an LDAP request has been sent.

time has a maximum value of 3600 seconds, and a value of 0, which is the minimum value as well as the default value, means that the wait is unlimited.

OAMLdapResponseWarningTime=threshold

If a connection to an LDAP server took longer than the threshold number of seconds specified by the **OAMLdapResponseWarningTime** parameter, then an [AMQ5544W](#) message will be written to the error log. The default threshold is 10 seconds.

ExpiryInterval

A time interval in seconds. Indicates the frequency with which the queue manager scans the queues looking for expired messages that have not already been cleaned up by other queue activities.

By default, the expiry scanner runs approximately every five minutes.



CAUTION: Altering the **ExpiryInterval** value is not typically required, and you should modify this value only under the guidance of IBM Support.

LivenessHeartBeatLen

Configures the frequency with which the queue manager checks that writes to the log are being made at a reasonable rate. The maximum value for **LivenessHeartBeatLen** is 600 seconds (10 minutes) and the minimum value is 0, which has the effect of disabling the check altogether.



CAUTION: In most cases, it is not necessary to change the frequency of these checks. Do not make any changes unless you are advised to do so by IBM Support.

ECHeartBeatLen

Configures the frequency of the queue manager general health checks. The minimum value for **ECHeartBeatLen** is 10000 milliseconds (10 seconds) and the maximum value is 60000 milliseconds (60 seconds).



CAUTION: In most cases, it is not necessary to change the frequency of these checks. Do not make any changes unless your are advised to do so by IBM Support.

FileLockHeartBeatLen

Changes the default value for the file lock checks for a multi-instance queue manager that the execution controller periodically performs to make sure that it still holds the exclusive lock on the primary multi-instance file. By default, these file lock checks are made every 20 seconds. The minimum value for **FileLockHeartBeatLen** is 10 seconds, and the maximum value is 600 seconds (10 minutes).



CAUTION: In most cases, it is not necessary to change the frequency of these checks. Do not make any changes unless your are advised to do so by IBM Support.

Multi

EventMinDuplicationDelayTime

The amount of time, in seconds, before a duplicate event should be put to the event queue. Any duplicate events that are attempted to be created within this time are discarded. A value of -1 means indefinite and a value of 0 means disabled, turning off rate limiting. The minimum value is -1 and the maximum value is 9999.

Example stanza

```
TuningParameters:
  SuppressDspAuthFail=NO
  ImplSyncOpenOutput=2
  OAMLdapConnectTimeout=60
  OAMLdapQueryTimeLimit=60
  OAMLdapResponseWarningTime=10
  ExpiryInterval=300
```

Related concepts

[Implicit syncpoint](#)

[Limiting the rate of positive authority events](#)

Multi

Variables stanza of the qm.ini file

The Variables stanza specifies configuration variables for use with automatic uniform clusters.

You can use attributes listed in the Variables stanza during automatic cluster configuration of CONNAME and the channel name MQSC fields of a cluster receiver channel. Configuration variables cannot be used in any other element of an MQSC script.

attribute=value

Specifies a name and associated value for use as an insert during MQSC definitions.

The *attribute=value* pairs can be supplied using the **-iv** command line option on the [crtmqm](#) command when you create a queue manager.

Example stanza

```
Variables:
  CONNAME=127.0.0.1(1414)
```

Related concepts

[“Automatic application balancing” on page 425](#)

Automatic application balancing greatly enhances application distribution and availability by enabling an IBM MQ uniform cluster to closely manage application distribution across the cluster, rather than relying on randomization, or a manual pinning of applications to specific queue managers.

Related tasks

[“Creating a new uniform cluster” on page 438](#)

How you create a new uniform cluster.

Related reference

[“Using automatic cluster configuration” on page 442](#)

You configure IBM MQ to enable automatic configuration by changing the `qm.ini` configuration information.

XAResourceManager stanza of the qm.ini file

The XAResourceManager stanza specifies information about the resource managers involved in global units of work coordinated by the queue manager.

Use the XAResourceManager stanza in the `qm.ini` file, to specify the information about the resource managers involved in global units of work coordinated by the queue manager.

  Alternatively, on Linux (x86 and x86-64) and Windows, use the IBM MQ Explorer the XA resource manager queue manager properties page.

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

Windows This attribute is mandatory 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.

Example stanza

```
XAResourceManager:
  Name=DB2 Resource Manager Bank
  SwitchFile=/usr/bin/db2swit
  XAOpenString=MQBankDB
  XACloseString=
  ThreadOfControl=THREAD
```

Note: 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.

IBM i Example qm.ini file for IBM i

An example showing 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.
SvrSndBufSize=20000       ; Size in bytes of the TCP/IP send buffer for each
                           ; channel instance. Default is 32768.
SvrRcvBufSize=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

TuningParameters:
ImplSyncOpenOutput=2

```

ALW Installation configuration file, mqinst.ini

On AIX and Linux systems, the installation configuration file, `mqinst.ini`, contains information about all the IBM MQ installations. On Windows systems, installation configuration information is in the registry.

Location of the mqinst.ini file

Linux

AIX

The `mqinst.ini` file is in the `/etc/opt/mqm` directory on AIX 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

Important: The `mqinst.ini` file must not be edited or referenced directly since its format is not fixed, and could change.

The installation identifier, for internal use only, is set automatically and must not be changed.

Instead of editing the `mqinst.ini` file directly, you must use the following commands to create, delete, query, and modify, the values in the file:

[`crtmqinst`](#) to create entries.
[`dlmqinst`](#) to delete entries.
[`dspmqinst`](#) to display entries.
[`setmqinst`](#) to set entries.

Installation configuration information on Windows

Windows

There is no `mqinst.ini` file on Windows. Installation configuration information is in the registry, and is held in the following key:

```
HKLM\SOFTWARE\IBM\WebSphere MQ\Installation\InstallationName
```

Important: This key must not be edited or referenced directly since its format is not fixed, and could change.

Instead, you must 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.

Multi IBM MQ MQI client configuration file, mqclient.ini

You can 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 configure your IBM MQ MQI clients by using a text file, similar to the queue manager configuration file, `qm.ini`. 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:

- **ALW** AIX, 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 173](#).

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 that use 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 175](#).

The configuration features apply to all connections that 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).

Here is an example of a 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=0
  ClntRcvBuffSize=0
  Connect_Timeout=0
```



```

MessageBuffer:
  MaximumSize=-1
  Updatepercentage=-1
  PurgeTime=0

LU62:
  TPName
  Library1=DLLName1
  Library2=DLLName2

PreConnect:
  Module=myMod
  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)

Connection:
  ApplName=ExampleApplName

```

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

Environment variables that were supported in releases earlier than IBM WebSphere MQ 7.0 continue to be supported in later releases, 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.

Comments in the configuration file



You can use the semicolon ';' and hash '#' character to mark the start of a comment within the configuration file. This can mark an entire line as a comment, or denote a comment at the end of a line which will not be included in the value of a setting.

If a value requires either of these characters, then you must escape that character using the backslash character '\'.

The following example shows the usage of comments within the configuration file:

```

# Example of an SSL stanza with comments
SSL:
  ClientRevocationChecks=REQUIRED ; Example of an end of line comment
  SSLCryptoHardware=GSK_PKCS11=/driver\;label\;password\;SYMMETRIC_CIPHER_ON # Example of
  escaped comment characters.

```

Related concepts

[The IBM MQ classes for Java configuration file](#)

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.

If the file is not present in location 1, then location 2 is tried, and so on.

1. **The location specified by the environment variable MQCLNTCF.**

You can choose a filename other than `mqclient.ini`. You can choose any valid path on your system that is readable by the client at run-time.

  These are examples of valid settings for MQCLNTCF on AIX and Linux systems when your file is located at `/data/mqclient.ini`.

```
export MQCLNTCF=/data/mqclient.ini
export MQCLNTCF=file:/data/mqclient.ini
export MQCLNTCF=file:///data/mqclient.ini
export MQCLNTCF=file://localhost/data/mqclient.ini
```

 These are examples of valid settings for MQCLNTCF on Windows systems when your file is located at `C:\data\mqclient.ini`.

```
set MQCLNTCF=C:\data\mqclient.ini
set MQCLNTCF=file:C:\data\mqclient.ini
set MQCLNTCF=file:///C:\data\mqclient.ini
set MQCLNTCF=file://localhost/C:\data\mqclient.ini
```

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 current working directory of the application.**
 3. **A file called `mqclient.ini` in the IBM MQ data directory for AIX, Linux, and Windows systems.**

  On AIX and Linux systems, the directory is `/var/mqm`.

 On Windows systems, the data directory is the location you chose when installing IBM MQ for the first time. The default is `C:\ProgramData\IBM\MQ`. The location is stored in the Windows Registry. If you copied the IBM MQ executable files onto your system, rather than installing by using the IBM MQ installer program, there are no Registry entries for IBM MQ, so this step is skipped.

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.

Notes: The IBM MQ data directory does not exist in the following cases:

-  on IBM i
-  on z/OS
- where the client has been supplied with another product

 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)
```

 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.

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.

  For C clients on AIX 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. This location is not available for .NET clients.

For more examples of environment variables, see “Environment variables descriptions” on page 65.

Multi 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 16. 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
ReconDelay	An administrative option to configure reconnect delay for client programs that can auto-reconnect.	Yes	No	Yes	Yes	Yes

Table 16. 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
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
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						

Table 16. 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
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
Connection stanza						
ApplName	The application name specified in the client configuration file.	Yes	No	No	No	No
JMQI stanza						
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						

Table 16. 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
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
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

Table 16. 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
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						
 AllowedQuantumSafeAlgorithms	Specifies which quantum safe algorithms the queue manager supports as a TLS server or TLS client.	Yes (C clients)				

Table 16. 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
AllowOutboundSNI	Specifies whether SNI capable clients will set SNI to the target IBM MQ channel name to the remote system when initiating a TLS connection.  Attention:  Deprecated From IBM MQ 9.3.0 this property is deprecated. Use OutboundSNI instead.	Yes	Yes	Yes	No	No
AllowTLSV13	Whether a queue manager is able to use the TLS 1.3 CipherSpecs.	Yes (C/C++ clients)	No	No	No	No
v 9.0.0 CCDTHttpsCertificateValPolicy	Specifies the level of validation that the IBM MQ client library runs when it retrieves a CCDT files token from an HTTPS endpoint.	No	No	Yes	No	No

Table 16. 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
<u>CDPCheckExtensions</u>	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
<u>CertificateLabel</u>	The certificate label of the channel definition.	Yes	No	No	No	No
<u>CertificateValidationPolicy</u>	Determines the type of certificate validation used.	Yes	No	No	No	No
<u>ClientRevocationChecks</u>	Determines how certificate revocation checking is configured if the client connect call uses an SSL/TLS channel.	Yes	No	No	No	No
<u>EncryptionPolicySuiteB</u>	Determines whether a channel uses Suite-B compliant cryptography and what level of strength is to be used.	Yes	No	No	No	No

Table 16. 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
Environment Scope	Controls whether IBM MQ uses a single IBM Global Security Kit (GSKit) Environment for the entire process or a GSKit environment per connection.	Yes (C clients)	No	No	No	No
V 10.0.0 HTTPSKeyStore	Specifies the path to a PKCS #12 key repository that the IBM MQ client library can use as a truststore when it creates outgoing HTTPS connections, such as fetching a CCDT file to use to connect to a queue manager.	Yes	No	No	No	No
V 10.0.0 HTTPSCertValidation	Specifies the level of validation that the IBM MQ client library runs when it makes an HTTPS connection.	Yes	No	No	No	No

Table 16. 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
HTTPSCertRevocation	Specifies the level of revocation checking that the IBM MQ client library runs when it makes an HTTPS connection, if the certificate that is used to establish trust for the connection provides a certificate revocation URL.	Yes	No	No	No	No
MinimumRSAKeySize	Specifies the minimum keysize that RSA certificates must have in order to be accepted.	Yes (C/C++ clients)	No	No	No	No
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

Table 16. 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
OCSPCheckExtensions	Controls whether IBM MQ acts on AuthorityInfo Access certificate extensions.	Yes	No	No	No	No
OCSPTimeout	The number of seconds to wait for an OCSP responder when performing a revocation check.	Yes	No	No	No	No
OutboundSNI	Specifies whether SNI capable clients will set SNI to the target IBM MQ channel name to the remote system when initiating a TLS connection, or to the hostname.	Yes	Yes	Yes	Yes	No
 PeerCertChainValidation	The GSKit certificate validation setting.	Yes	No	No	No	No
  RFC5280EKUChecks	Validates the Extended Key Usage (EKU) extension of the server certificate.	Yes (C clients)				

Table 16. 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
<u>SSLCryptoHardware</u>	Sets the parameter string required to configure PKCS #11 cryptographic hardware present on the system.	Yes	No	No	No	No
<u>SSLCryptoHardwareKeyFile</u>	Specifies the full path and name of the file containing the initial key that was used to encrypt the password in the PKCS #11 cryptographic hardware configuration string that is specified with the SSLCryptoHardware attribute.	Yes	No	No	No	No

Table 16. 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
SSLFipsRequired	 Specifies whether a FIPS certified cryptographic library and algorithms are to be used if cryptography is carried out in IBM MQ.  Specifies whether only FIPS certified algorithms are to be used if cryptography is carried out in IBM MQ.	Yes	No	No	No	No
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
SSLHTTPConnectTimeout	The number of seconds to wait for a network connection to be successfully established to a HTTP server when performing a revocation check.	Yes	No	No	No	No

Table 16. 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
SSLKeyRepository	The location of the key repository that holds the user's digital certificate, in stem format.	Yes	No	No	No	No
SSLKeyRepositoryPassword	The passphrase to access the key repository.	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
V 10.0.0 TokenHttpsCertificateValPolicy	Specifies the level of validation that the IBM MQ client library runs when it retrieves a JWT token from an HTTPS endpoint.	No	No	Yes	No	No
TCP stanza						
ClnRcvBufferSize	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 16. 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
CIntSndBufferSize	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
 Library1	On Windows only, the name of the TCP/IP sockets DLL.	Yes	No	No	No	No
  DNSResolutionOrder	The order in which IBM MQ looks up host names in mixed IPv6 and IPv4 environments.	Yes				
Trace stanza Note: The Trace stanza applies to the IBM MQ .NET and XMS .NET clients only.						

Table 16. 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
MQDotnetTraceLevel	Used to enable IBM MQ .NET trace.	No	No	No	Yes	No
MQDotnetTracePath	Points to a folder where the IBM MQ .NET trace files will be created.	No	No	No	Yes	No
MQDotnetErrorPath	Points to a folder where error log files will be created for IBM MQ .NET trace.	No	No	No	Yes	No
XMSDotnetTraceLevel	Used to enable XMS .NET trace.	No	No	No	No	Yes
XMSDotnetTraceFilePath	Points to a folder where the XMS .NET trace files will be created.	No	No	No	No	Yes
XMSDotnetTraceSpecification	Specifies the name of the class you want to trace for XMS .NET.	No	No	No	No	Yes
XMSDotnetTraceSpecification	Specifies the maximum size of the trace file that should be generated for XMS .NET.	No	No	No	No	Yes
XMSDotnetTraceFileSize	The number of trace files that are to be retained for XMS .NET.	No	No	No	No	Yes

Application stanza of the client configuration file

Use the Application stanza to specify attributes which influence the uniform cluster balancing behavior for a specific application connecting using this configuration. Values in this stanza take precedence over the ApplicationDefaults stanza, but can be overridden by an MQBNO structure provided through a program.

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 Application stanza:

Name = ApplicationName

Identifies which application name the options apply to.

Type = Simple,ReqRep

Indicates to IBM MQ the general pattern of IBM MQ activity in which this application participates.

BalanceTimeout = Never,Immediate,0-999999999,Default

Indicates to IBM MQ the timeout before application activity can be interrupted to permit re-balancing; either never, or a value up to a maximum of 999,999,999 seconds, with a default value of 10 seconds.

BalanceOptions = None,IgnTrans

Either no balancing options are set or permits immediate interruption of applications currently involved in a transaction.

ApplicationDefaults stanza of the client configuration file

Use the ApplicationDefaults stanza to specify attributes which influence the default uniform cluster balancing behavior for client applications which connect using this configuration. These defaults can be overridden by either an application specific Application stanza, or an MQBNO structure provided through a program.

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 ApplicationDefaults stanza:

Type = Simple,ReqRep

Indicates to IBM MQ the general pattern of IBM MQ activity in which this application participates.

BalanceTimeout = Never,Immediate,0-999999999,Default

Indicates to IBM MQ the timeout before application activity can be interrupted to permit re-balancing; either never, or a value up to a maximum of 999,999,999 seconds, with a default value of 10 seconds.

BalanceOptions = None,IgnTrans

Either no balancing options are set or permits immediate interruption of applications currently involved in a transaction.

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 variable.

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.

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

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

ChannelDefinitionDirectory can contain a URL which works in combination with the ChannelDefinitionFile attribute (see [“URL access to the CCDT”](#) on page 53).

The ChannelDefinitionDirectory path is equivalent to the **MQCHLLIB** environment variable.

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 variable.

The maximum length of the combined ChannelDefinitionDirectory and ChannelDefinitionFile attributes is the maximum file path length of the operating system.

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

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

Automatic client reconnection usually depends on two values which are the:

- Reconnect option set in the application MQCONN (or JMS Connection Factory)
- Default reconnect option supplied in any client connection definition in use (MQCD structure, for example supplied using a CCDT file).

The mqclient.ini file attribute applies **only** if there is no channel definition being used that sets the **DefReconnect** attribute and in that situation behaves as though one had been supplied. The channel **DefReconnect** attribute (and therefore this attribute, if applicable):

- Override the application code if either is set to DISABLED
- Are overridden by the application code in all other cases, if the options are specified on MQCONN

See the description of DEFRECON for a table showing all possible combinations of application and channel definition supplied values.

Notes:

- If an MQCD is in use but predates MQCD_VERSION_10, the **DefReconnect** parameter is not part of the structure. In this situation, the value of that missing parameter is populated with the `mqclient.ini` file **DefReconnect** value if one is specified. This can occur, for example, if a binary format CCDT generated in an older version of IBM MQ is still in use by a client application.
- When interpreted by the IBM MQ client code, a JSON CCDT, see [“Configuring a JSON format CCDT” on page 45](#), always generates MQCD structures at the latest version, and therefore always supplies the default value (NO) for this attribute unless explicitly present with a different value.

MQReconnectTimeout

The maximum time, in seconds, that the automatic client reconnect function in a client tries to reestablish the connection. 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 variable 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 a TLS 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 (default) | 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, 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 (default) |always|optional

From IBM MQ 8.0, authentication credentials that IBM MQ client applications specify when they connect to a queue manager can be protected by using the IBM MQ MQCSP password protection feature, if the connection does not use TLS encryption.

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.

MQCSP password protection is useful for test and development purposes as using MQCSP password protection is simpler than setting up TLS encryption, but not as secure.

For more information about protecting credentials in the MQCSP structure, and the values that can be set for this attribute, 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, and managed XMS .NET clients. IBM MQ classes for Java and IBM MQ classes for JMS clients can also read this attribute but do not support 32-bit channel exits that are not written in Java. If you attempt such a scenario, connection fails and an error is generated.

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 MQ system property.

Connection stanza of the client configuration file

Use the connection stanza to specify an application name.

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 Connection stanza:

ApplName = ExampleApplName

You can specify an application name in the client configuration file.

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

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 in versions earlier than IBM MQ 9.3.0.

YES

Use MQCSP authentication mode when authenticating with a queue manager. This is the default value from IBM MQ 9.3.0.

There are several other ways to set the authentication mode that take precedence over the value of the **useMQCSPauthentication** attribute. For more information about Compatibility mode and MQCSP authentication mode, see [Connection authentication with the Java client](#).

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 variable.

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 through 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 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 through 999 999 seconds, or the special value 0, meaning that no purge takes place.

UpdatePercentage = integer|-1

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 a new request is sent to the server.

The threshold *T* is calculated as follows:

$$T = \text{Upper} - \text{Lower}$$

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 through 100, and has a default of 20. Lower is calculated as follows:

$$\text{Lower} = \text{Upper} \times \text{UpdatePercentage} / 100$$

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

$$T = \text{Upper} - \text{Lower}$$

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 attributes 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

$$T = \text{Upper} - \text{Lower}$$

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.

Multi

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 = *user_data*

This attribute specifies user data that is passed to the preconnect exit. The data that is passed to the preconnect exit is specific to the implementation of the preconnect exit that you are using and what data it is expecting to be passed.

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

For example, this attribute could be used to specify the URL of the repository where connection definitions are stored, such as, 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 = *myMod*

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 an IBM MQ client to connect to a queue manager from an earlier version of the product and a 2085 (MQRC_UNKNOWN_OBJECT_NAME) error is reported.

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.

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.

MQIInitialKeyFile = *pathname*

The full path and name of the file containing the initial key that was used to encrypt credentials provided by the client. The initial key must be specified if an initial key file was specified when the key repository passphrase was encrypted using the **runmqicred** utility.

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

AllowedQuantumSafeAlgorithms=ALL (default) | NONE | HYBRIDMLKEM | MLKEM

Specifies which quantum safe algorithms the application supports as a TLS client. This can be a comma separated list of algorithms or a single special value of ALL or NONE. Duplicates or invalid values will result in an error and a default setting. The default is ALL.

- ALL: Enables all quantum safe algorithms.
- NONE: Disables all quantum safe algorithms.
- HYBRIDMLKEM: Enables the Hybrid ML-KEM algorithm.
- MLKEM: Enables the ML-KEM algorithm.

Deprecated **AllowOutboundSNI = YES (default) | NO**

If enabled, SNI capable clients will set SNI to the target IBM MQ channel name to the remote system when initiating a TLS connection. If this attribute is set to NO, SNI capable clients will not set the SNI header causing outbound connection requests to receive the remote queue manager's default certificate during the TLS handshake, and so per-channel certificates cannot be used.

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

Property values are interpreted case-sensitively by the Java/JMS client, so YES/NO values should be set in uppercase.



Attention: **Deprecated** From IBM MQ 9.3.0 the **AllowOutboundSNI** property is deprecated, and is available for backwards-compatibility purposes only.

AllowOutboundSNI set to YES provides the same function as **OutboundSNI** set to CHANNEL, while **AllowOutboundSNI** set to NO provides the same function as **OutboundSNI** set to HOSTNAME.

If both **AllowOutboundSNI** and **OutboundSNI** attributes are present in the SSL stanza, the setting of **OutboundSNI** takes precedence.

IBM I **ALW** **AllowTLSV13=Y | YES | T | TRUE (default) | N | NO | F | FALSE**

Specifies whether a queue manager is able to use the TLS 1.3 CipherSpecs (see [Enabling CipherSpecs](#)).

This attribute can be read by C/C++ clients.

This attribute has the following possible values:

- Y (default) , YES (default) , T (default), or TRUE (default): Enables TLS 1.3 which allows the queue manager to use the TLS 1.3 CipherSpecs.
- N, NO, F, or FALSE: Disables TLS 1.3, which means that the queue manager is unable to use the TLS 1.3 CipherSpecs.

LTS

Note: When using the MQI client, the value of **AllowTLSV13** is inferred unless it is explicitly specified in the SSL stanza of the “[SSL stanza of the client configuration file](#)” on page 198 file being used by the application. For more information, see [IBM MQ MQI client and TLS 1.3](#).

V 10.0.0 **CCDTHttpsCertValPolicy = ANY | HOSTNAMECN (default) | NONE**

Specifies the level of validation that the IBM MQ client library runs when it retrieves a CCDT files token from an HTTPS endpoint.

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

These attributes have the following possible values:

ANY

Verifies that the peer's certificate is valid, signed by a trusted authority, and ignores hostname verification. If a customized SSLSocketFactory is provided, then it overrides the default mechanism of Java Secure Socket Extension (JSSE).

HOSTNAMECN (default)

Verifies that the peer's certificate is valid, signed by a trusted authority, and that the certificate distinguished name (including the CN and SAN extensions) matches that of the HTTP server being contacted. If a customized SSLSocketFactory is provided, then it overrides the default mechanism of Java Secure Socket Extension (JSSE).

NONE

All server certificates are accepted without validation. An internal handler implementation that provides an SSLSocketFactory instance trusting all certificates will be used. If a customized SSLSocketFactory is provided, it overrides the handler implementation.

Note: If a client application does not supply any values, the default value that IBM MQ classes for JMS performs is HOSTNAMECN.

CDPCheckExtensions = YES|NO (default)

CDPCheckExtensions specifies whether 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: TLS channels try to check CDP servers to determine whether a digital certificate is revoked.
- NO: 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.

See [Certificate label \(CERTLABL\)](#) for more information.

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.

NONE

Use no certificate validation.

ClientRevocationChecks = REQUIRED|OPTIONAL|DISABLED

Determines how certificate revocation checking is configured if the client connect call uses a 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.

If you are using either MQSCO or CCDT, then the connection succeeds. If there is no CCDT file and if MQSCO is also not supplied, then the connection fails with a reason code 2059 and the error log reports AMQ9518E: File '/var/mqm/AMQCLCHL.TAB' not found.

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.

ALW EnvironmentScope=PROCESS|CONNECTION

Controls whether IBM MQ uses a single IBM Global Security Kit (GSKit) Environment for the entire process or a GSKit environment per connection.

This attribute can be read by C clients.

This attribute has the following possible values:

PROCESS

A single GSKit environment is used for multiple connections created by the process. Using this setting means that TLS keystore changes will not be available until all active TLS connections within the process have stopped.

This value is the default value.

CONNECTION

A GSKit environment is created for each connection within the same process. Enabling this means TLS keystore changes will be picked up immediately by any new TLS connections started by the process.



Warning: Enabling this mode of operation causes applications to use additional CPU and Memory resources to create each GSKit environment. This resource consumption increases with each additional concurrent TLS connection.

V 10.0.0 HTTPSKeyStore = *string*

Specifies the path to a PKCS #12 key repository that the IBM MQ client library can use as a truststore when it creates outgoing HTTPS connections, such as fetching a CCDT file to use to connect to a queue manager. The key repository file must be encrypted and accompanied by a stash file (a file with a .sth extension) of the same name, which is used when the client library needs to access the key repository.

This attribute does not have a default value. If you do not provide a value, function that requires an HTTPS truststore in the IBM MQ client is disabled.

This attribute can be read by C and unmanaged .NET clients. If you update this attribute, you must restart the client application to begin using the key repository file that you specified.

See [Creating a key repository for use as a TLS truststore](#) for information about creating a truststore that uses certificates from AIX and Linux operating systems.

HTTPSCertValidation = ANY (default) | HOSTNAMECN | NONE

Specifies the level of validation that the IBM MQ client library runs when it makes an HTTPS connection.

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

This attribute has the following possible values:

ANY (default)

Check that the peer's certificate is valid and is signed by a trusted authority.

HOSTNAMECN

Check that the peer's certificate is valid and is signed by a trusted authority, and that the certificate distinguished name matches that of the HTTP server being contacted.

NONE

Do not check the peer's certificate. The session is encrypted with TLS but there is no guarantee that the client is talking to the intended host. Use this option with caution; use of this option could, for example, enable a man-in-the-middle attack that redirects the client to a malicious CCDT file.

HTTPSCertRevocation = REQUIRED (default) | OPTIONAL | DISABLED

Specifies the level of revocation checking that the IBM MQ client library runs when it makes an HTTPS connection, if the certificate that is used to establish trust for the connection provides a certificate revocation URL.

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

This attribute has the following possible values:

REQUIRED (default)

Check the certificate's revocation status against the specified server. If the server cannot be reached, assume that the certificate is invalid and reject the HTTPS connection.

OPTIONAL

Check the certificate's revocation status against the specified server. If the revocation check cannot be completed for some reason (for example, the revocation server cannot be reached) accept the certificate as valid.

DISABLED

Do not run certificate revocation checks.

ALW MinimumRSAKeySize=int

Specifies the minimum keysize that RSA certificates must have in order to be accepted. Allows any value equal to 0 or above. Defaults to **LTS** 1 or **V 10.0.0** 2048 if not specified.

This attribute can be read by C/C++ clients.

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.

If no OCSP responder is defined, then OCSP revocation checks are not performed and the connection is allowed to continue.

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 MQRCQ_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 MQRCQ_SSL_UNKNOWN_REVOCATION is generated. The connection is allowed to continue.

OCSPCheckExtensions = YES|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.

ALW **OCSPTimeout = *number***

The number of seconds to wait for an OCSP responder when performing a revocation check.

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

From IBM MQ 9.3.0, if a value of 0 is set, the default timeout of 30 seconds is used.

If no value is set, then the IBM MQ default of 30 seconds is used.

OutboundSNI = CHANNEL | HOSTNAME

If **OutboundSNI** is set to CHANNEL, SNI capable clients set SNI to the target IBM MQ channel name to the remote system when initiating a TLS connection.

If this attribute is set to HOSTNAME, SNI capable clients set the SNI header to the hostname, causing outbound connection requests to receive the default certificate of the remote queue manager during the TLS handshake, and so per-channel certificates cannot be used.

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

Property values are interpreted case-sensitively by the Java/JMS client, so YES/NO values should be set in uppercase.

From IBM MQ 9.3.0, the IBM MQ managed .NET client has been updated to set SERVERNAME to the respective hostname if the **OutboundSNI** property is set to HOSTNAME, which allows an IBM MQ managed .NET client to connect to a queue manager using [Red Hat OpenShift routes](#).

Note: If an application with an **OutboundSNI** setting of HOSTNAME connects to a channel with a certificate label configured, the application is rejected with an MQRC_SSL_INITIALIZATION_ERROR, and an AMQ9673 message is printed in the queue manager error logs.

ALW **PeerCertChainValidation=*string***

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

The string can be one of two values:

- Usepeerchain **[Default]**: The certificate chain provided by the peer can be used to bridge any trust chain gaps while validating certificates. With the exception of the root certificate.
- Truststoreonly **[Not recommended]**: Only certificates in the trust store will be used for validating the peer's certificate.

ALW **RFC5280EKUChecks = TRUE | FALSE**

Validates the Extended Key Usage (EKU) extension of the server certificate.

This attribute can be read by C clients.

This attribute has the following possible values:

TRUE

If the queue manager certificate has specified an EKU that does not include `serverAuth`, then the connection is rejected. This is the default value.

FALSE

The EKU is not validated whether the queue manager certificate has specified an EKU or not.

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. The value of this parameter is either:

SYMMETRIC_CIPHER_OFF

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

SYMMETRIC_CIPHER_ON

Access symmetric cipher operations.



When supplying the different components of the string, you must escape the semicolon characters using the backslash character, as the semicolon character is treated as a comment. For example: `'\;'`

You should protect the token password contained in the **SSLCryptoHardware** attribute string. For more information, see [IBM MQ clients using cryptographic hardware](#).

To handle encrypted passwords, there is now no limit to the length of the string.

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

SSLCryptoHardwareKeyFile = *pathname*

The full path and name of the file containing the initial key that was used to encrypt the password in the PKCS #11 cryptographic hardware configuration string that is specified with the **SSLCryptoHardware** attribute. The initial key must be specified if an initial key file was specified when the password in the cryptographic hardware configuration string was encrypted using the **runp11cred** command. For more information, see [IBM MQ clients using cryptographic hardware](#).

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

SSLFipsRequired = YES|NO

 Specifies whether a FIPS certified cryptographic library and 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.



Attention:  FIPS support is currently not available for Linux s390x. Enabling FIPS on a application will prevent all TLS and AMS related operations. Customers on this platform who require FIPS support should remain at a previous version of IBM MQ. Support for FIPS 140-3

will be enabled in a future FixPack once the IBM Crypto for C (ICC) cryptographic module has received its certification on this platform.

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.

AIX For 32-bit clients on AIX, the network address can be only an IPv4 address.

On other platforms the network address 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.

SSLHTTPConnectTimeout = *number|0*

The number of seconds to wait for a network connection to be successfully established to a HTTP server when performing a revocation check.

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

If no value is set, then the IBM MQ default of 0 (off) is used.

SSLKeyRepository = *pathname*

The full path and the file name of the key repository that holds the user's digital certificate.

If the file extension is not specified:

- **V 10.0.0** IBM MQ first checks for a file with extension .p12 before checking for a file with extension .kdb.
- **LTS** If the file suffix is not specified, it is assumed to be .kdb.

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

SSLKeyRepositoryPassword = *passphrase*

The passphrase to access the key repository. The value can be a plain text string, or a passphrase that has been encrypted using the **runmqicred** utility.

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

SSLKeyResetCount = *integer|0*

The number of unencrypted bytes sent and received on a 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, 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.

V 10.0.0 TokenHttpsCertValPolicy = ANY|HOSTNAMECN (default)|NONE

Specifies the level of validation that the IBM MQ client library runs when it retrieves a JWT token from an HTTPS endpoint.

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

These attributes have the following possible values:

ANY

Verifies that the peer's certificate is valid, signed by a trusted authority, and ignores hostname verification. If a customized SSLSocketFactory is provided, then it overrides the default mechanism of Java Secure Socket Extension (JSSE).

HOSTNAMECN (default)

Verifies that the peer's certificate is valid, signed by a trusted authority, and that the certificate distinguished name (including the CN and SAN extensions) matches that of the HTTP server being contacted. If a customized SSLSocketFactory is provided, then it overrides the default mechanism of Java Secure Socket Extension (JSSE).

NONE

All server certificates are accepted without validation. An internal handler implementation that provides an SSLSocketFactory instance trusting all certificates will be used. If a customized SSLSocketFactory is provided, it overrides the handler implementation.

Note: If a client application does not supply any values, the default value that IBM MQ classes for JMS performs is HOSTNAMECN.

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:

ClnRcvBuffSize = *number*|0

The size (in bytes) of the TCP/IP receive buffer that is used by the client end of a client-connection or server-connection channel.

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

Value	Behavior
Not set  (default)	When creating a TCP/IP connection, the client queries the network layer for the current receive buffer size then modifies the size as follows: • Current receive buffer size is 32768 bytes or larger: the client lets the network layer manage the buffer size dynamically. The actual buffer size is logged in message JMSCS0071 in the IBM MQ classes for Java, IBM MQ classes for JMS, or IBM MQ classes for Jakarta Messaging log file, in the IBM MQ classes for .NET client trace file or IBM MQ classes for XMS .NET trace file. • Current receive buffer size is less than 32768 bytes: the client requests the network layer to set the buffer size to 32768 bytes. The actual buffer size is logged in message JMSCS0072 in the IBM MQ classes for Java, IBM MQ classes for JMS, or IBM MQ classes for Jakarta Messaging log file, in the IBM MQ classes for .NET client trace file or IBM MQ classes for XMS .NET trace file.
0	The network layer manages the buffer size dynamically.
A positive integer	The client requests the network layer to use the specified value as the receive buffer size.

Notes:

-  In previous releases, C clients had the same default value and behavior. Other clients had a default value of 32768 bytes.
- The requested buffer size is not guaranteed. The network layer might use a different size if the requested value is unsuitable.
- Operating systems vary in how they determine the buffer size in the network layer; their methods are typically described in the **tcp** or `getsockopt()` and `setsockopt()` manual pages for the operating system.

ClntSndBuffSize = *number*|0

The size (in bytes) of the TCP/IP send buffer that is used by the client end of a client-connection or server-connection channel.

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

Value	Behavior
Not set V 3000 (default)	When creating a TCP/IP connection, the client queries the network layer for the current send buffer size then modifies the size as follows: • Current send buffer size is 32768 bytes or larger: the client lets the network layer manage the buffer size dynamically. The actual buffer size is logged in message JMSCS0069 in the IBM MQ classes for Java, IBM MQ classes for JMS, or IBM MQ classes for Jakarta Messaging log file, in the IBM MQ classes for .NET client trace file or IBM MQ classes for XMS .NET trace file. • Current send buffer size is less than 32768 bytes: the client requests the network layer to set the buffer size to 32768 bytes. The actual buffer size is logged in message JMSCS0070 in the IBM MQ classes for Java, IBM MQ classes for JMS, or IBM MQ classes for Jakarta Messaging log file, in the IBM MQ classes for .NET client trace file or IBM MQ classes for XMS .NET trace file.
0	The network layer manages the buffer size dynamically.
A positive integer	The client requests the network layer to use the specified value as the send buffer size.

Notes:

- V 3000 In previous releases, C clients had the same default value and behavior. Other clients had a default value of 32768 bytes.
- The requested buffer size is not guaranteed. The network layer might use a different size if the requested value is unsuitable.
- Operating systems vary in how they determine the buffer size in the network layer; their methods are typically described in the **tcp** or `getsockopt()` and `setsockopt()` manual pages for the operating system.

Connect_Timeout = *number*

The maximum number of seconds before an attempt to connect the socket times out.

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

If a nonzero value is given for **Connect_Timeout**, IBM MQ asks the operating system to wait a maximum of this number of seconds before returning the failure condition.

If the local operating system is told immediately by the remote side that nothing is listening on the port, the connection attempt normally fails immediately, regardless of any value you set for **Connect_Timeout**. (On Windows, the operating system normally takes 1-2 seconds to notify IBM MQ about this condition).

If the local operating system receives nothing from the remote side (this happens if you specified an address for a machine that is offline, or does not exist) then normally **Connect_Timeout** seconds elapse before the operating system notifies IBM MQ of the failure condition. If you specify zero for **Connect_Timeout** (this is the default), the maximum time used is 20 seconds.

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** and the **MQIPADDRV** environment variable.

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.

Library1 = DLLName|WSOCK32

(Windows only) The name of the TCP/IP sockets DLL.

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

DNSResolutionOrder=list|IPV6, IPV4, ANY(default)

The order in which IBM MQ looks up host names in mixed IPv6 and IPv4 environments. If the **AMQ_NO_IPV6** environment variable is set, IBM MQ acts as if **DNSResolutionOrder=IPV4** is set, no matter what value is configured.

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

The value of **DNSResolutionOrder** is a comma-separated list with the values IPV6, IPV4, and ANY in the order preferred. These values correspond to IPv6, IPv4 and unspecified address family DNS lookups, where an unspecified lookup can resolve both IPv6 and IPv4 addresses. Each value can appear in the list at most once. If IPV4 or IPV6 are missing from the list, IBM MQ implicitly appends ANY to make sure it can resolve all address types.

Setting the list value to just ANY might give faster performance. IPV6 and IPV4 can be listed in the order reflecting their importance on the local network.

Trace stanza of the client configuration file

Use the Trace stanza to enable trace for the IBM MQ .NET and XMS .NET client libraries.

The following attributes can be included in the TRACE stanza:

MQDotnetTraceLevel=0 (default) |1|2

The **MQDotnetTraceLevel** attribute is used to start or stop IBM MQ .NET trace:

- 0: Stops tracing - this is the default value.
- 1: Starts tracing with fewer details.
- 2: Starts tracing with full details - recommended.

This attribute can be read by the managed IBM MQ .NET client.

MQDotnetTracePath=pathname

The **MQDotnetTracePath** attribute points to a folder where the IBM MQ .NET trace files will be created. The current directory of the application is used if the path is blank or the **MQDotnetTracePath** property is not defined.

This attribute can be read by the managed IBM MQ .NET client.

MQDotnetErrorPath=pathname

The **MQDotnetErrorPath** attribute points to a folder where error log files will be created for IBM MQ .NET trace. The current directory of the application is used if the path is blank or if the **MQDotnetErrorPath** attribute is not defined.

This attribute can be read by the managed IBM MQ .NET client.

XMSDotnetTraceLevel=0 (default) |1|2

The **XMSDotnetTraceLevel** attribute is used to start or stop XMS .NET trace:

- 0: Stops tracing - this is the default value.
- 1: Starts tracing with basic format.

- 2: Starts tracing with Advanced format.

This attribute can be read by the managed XMS .NET client.

XMSDotnetTraceFilePath=filename

If a value is not set for the **XMSDotnetTraceFilePath** attribute, or if this attribute is present but contains an empty string, the trace file for XMS .NET is placed in the current directory. To store the trace file in a named directory, specify the directory name in the **XMSDotnetTraceFilePath**, for example `XMSDotnetTraceFilePath="c:\somepath"`.

This attribute can be read by the managed XMS .NET client.

XMSDotnetTraceSpecification=ComponentName=type=state

The **XMSDotnetTraceSpecification** attribute specifies the name of the class you want to trace and the type of trace you require for XMS .NET:

- *ComponentName* is the name of the class that you want to trace. You can use a * wildcard character in this name. For example, `*=all=enabled` specifies that you want to trace all classes, and `IBM.XMS.impl.*=all=enabled` specifies that you require API trace only.
- *type* can be any of the following trace types: all, debug, event, EntryExit.
- *state* can be either enabled or disabled.

You can string multiple trace elements together by using a ':' (colon) delimiter.

This attribute can be read by the managed XMS .NET client.

XMSDotnetTraceFileSize=size

The **XMSDotnetTraceFileSize** attribute specifies the maximum size of the trace file that should be generated for XMS .NET. The default maximum is 20 MB, which is specified as `XMSDotnetTraceFileSize=20`.

This attribute can be read by the managed XMS .NET client.

XMSDotnetTraceFileNumber=number

The **XMSDotnetTraceFileNumber** attribute specifies the number of trace files that are to be retained for XMS .NET. The default is 4 (one active file and three archive files). The minimum number allowed is 2.

This attribute can be read by the managed XMS .NET client.

Related tasks

[Tracing IBM MQ .NET applications with mqclient.ini](#)

[Tracing XMS .NET applications with mqclient.ini](#)

Multi

Activity trace configuration file, mqat.ini

The activity trace configuration file, `mqat.ini`, is used to configure activity trace behavior. This file is used to define the level and frequency of reporting activity trace data. The file also provides a way to define rules to enable and disable activity trace based on the name of an application.

The `mqat.ini` file follows the same stanza key and parameter-value pair format as the `mqc.ini` and `qm.ini` files. The file consists of a single stanza, `AllActivityTrace`, which is used to configure the level and frequency of reporting activity trace data by default for all activity trace. The file can also contain multiple `ApplicationTrace` stanzas. Each one of these stanzas defines a rule for the trace behavior for one or more connections, based on matching the application name of the connections to the rule. For more information, see [Application activity trace](#) and [Configuring activity trace behavior using mqat.ini](#).

The queue manager applies a number of rules to determine which stanzas settings to use for a connection. Optionally, you can override the global trace level and frequency settings under the `AllActivityTrace` stanza for those connections that match an `ApplicationTrace` stanza. For more information, see [Configuring activity trace behavior using mqat.ini](#).

Directory locations

Linux **IBM i** **AIX** On AIX and Linux and IBM i systems, `mqat.ini` is located in the queue manager data directory, which is the same location as the `qm.ini` file.

Windows On Windows systems, `mqat.ini` is located in the queue manager data directory `C:\Program Files\IBM\WebSphere MQ\mqmgs\queue_manager_name`. Users running applications to be traced need permission to read this file.

Multi AllActivityTrace stanza of the mqat.ini file

The AllActivityTrace stanza of the `mqat.ini` configuration file specifies the parameters that are used to configure the trace levels for a queue manager.

A single AllActivityTrace stanza defines settings for the activity trace that is applied to all IBM MQ connections, unless overridden.

Individual values in the AllActivityTrace stanza can be overridden by more specific information in an ApplicationTrace stanza.

If more than one AllActivityTrace stanza is specified then the values in the last stanza is used. Parameters missing from the chosen AllActivityTrace take default values. Parameters and values from previous AllActivityTrace stanzas are ignored.

ActivityInterval

The time interval in seconds between trace messages. Activity trace does not use a timer thread, so the trace message is not written at the exact instant that the time elapses, it is written when the first MQI operation is executed after the time interval elapses. If this value is 0, the trace message is written when the connection disconnects (or when the activity count is reached). Defaults to 1.

ActivityCount

The number of MQI operations between trace messages. If this value is 0, the trace message is written when the connection disconnects (or when the activity interval elapses). Defaults to 100.

TraceLevel

The amount of parameter detail that is traced for each operation. The description of individual operations details which parameters are included for each trace level. Set to LOW, MEDIUM, or HIGH. Defaults to MEDIUM.

TraceMessageData

The amount of message data that is traced in bytes for MQGET, MQPUT, MQPUT1, and Callback operations. Defaults to 0.

StopOnGetTraceMsg

Can be set to ON or OFF. Defaults to ON.

SubscriptionDelivery

Can be set to BATCHED or IMMEDIATE. Determines whether the **ActivityInterval** and **ActivityCount** parameters are to be used when one or more activity trace subscriptions are present. Setting this parameter to IMMEDIATE results in the **ActivityInterval** and **ActivityCount** values being overridden with effective values of 1 when the trace data has a matching subscription. Each activity trace record is not batched with other records from the same connection and instead delivered to the subscription immediately with no delay. The IMMEDIATE setting increases the performance overhead of collecting activity trace data. The default setting is BATCHED.

Related tasks

[Configuring activity trace behavior using mqat.ini](#)

Multi ApplicationTrace stanza of the mqat.ini file

The `mqat.ini` configuration file can contain multiple `ApplicationTrace` stanzas. Each one these stanzas defines a rule for the trace behavior for one or more connections, based on matching the application name of the connections to the rule.

You can set the following values for the `ApplicationTrace` stanza:

Trace

Activity trace switch that can be set to ON or OFF. The **Trace** parameter is a required parameter with no default value. It can be used in the application-specific stanza to determine whether activity trace is active for the scope of the current application stanza. Note that this value overrides the **ACTVTRC** and **ACTVCONO** settings for the queue manager.

ApplName

The **ApplName** parameter is specified as a character string and is a required parameter with no default. This value is used to determine which applications the `ApplicationTrace` stanza applies to. It is matched to the **ApplName** value from the API exit context structure (which is equivalent to the `MQMD.PutApplName`). The content of the **ApplName** value varies according to the application environment.

On Multiplatforms, only the filename portion of the `MQAXC.ApplName` is matched to the value in the stanza. Characters to the left of the rightmost path separator are ignored when the comparison is made.

A single wildcard character (*) can be used at the end of the **ApplName** value to match any number of characters after that point. If the **ApplName** value is set to a single wildcard character (*) then the **ApplName** value matches all applications.

IBM i ApplFunction

The **ApplFunction** parameter is specified as a character string. The default value is *. The value of this parameter is used to qualify which application programs the `ApplicationTrace` stanza and the **ApplName** value applies to.

The stanza is optional, and is only valid for IBM i queue managers. A single wildcard character (*) can be used at the end of the **ApplName** value to match any number of characters. For example, an `ApplicationTrace` stanza specifying **ApplName** = * and **ApplFunction** = `AMQSPUT0` applies to all invocations of the `AMQSPUT0` program from any job.

ApplClass

The **ApplClass** parameter defines the class of an application and can be set to the following values:

- USER
- MCA
- ALL (This is the default value)

For an explanation of how the **AppType** values correspond to IBM MQ connections, see [Table 3 in Configuring activity trace behavior using mqat.ini](#).

Optionally, the global trace level and frequency settings under the `AllActivityTrace` stanza can be overridden for those connections matching an `ApplicationTrace` stanza.

The following parameters can be set under an `ApplicationTrace` stanza. If they are not set, the value is inherited from the [AllActivity trace stanza](#) settings:

ActivityInterval

The time interval in seconds between trace messages. Activity trace does not use a timer thread, so the trace message is not written at the exact instant that the time elapses, it is written when the first MQI operation is executed after the time interval elapses. If this value is 0, the trace message is written when the connection disconnects (or when the activity count is reached). Defaults to 1.

ActivityCount

The number of MQI operations between trace messages. If this value is 0, the trace message is written when the connection disconnects (or when the activity interval elapses). Defaults to 100.

TraceLevel

The amount of parameter detail that is traced for each operation. The description of individual operations details which parameters are included for each trace level. Set to LOW, MEDIUM, or HIGH. Defaults to MEDIUM.

TraceMessageData

The amount of message data that is traced in bytes for MQGET, MQPUT, MQPUT1, and Callback operations. Defaults to 0.

StopOnGetTraceMsg

Can be set to ON or OFF. Defaults to ON.

Related tasks

[Configuring activity trace behavior using mqat.ini](#)

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 you begin

Before reading this section it is helpful to have an understanding of channels, queues, and the other concepts introduced in [Distributed queuing and clusters](#).

If you need to connect two queue managers that are on different physical networks, or need to communicate through a firewall, using IBM MQ Internet Pass-Thru might simplify the configuration. For more information, see [IBM MQ Internet Pass-Thru](#).

Procedure

- Use the information in the following subtopics to connect your applications using distributed queuing:
 - [“IBM MQ distributed queuing techniques” on page 213](#)
 - [“Introduction to distributed queue management” on page 233](#)
 - [“How to send a message to another queue manager” on page 236](#)
 - [“Triggering channels” on page 257](#)
 - [“Safety of messages” on page 255](#)
 -  [“Monitoring and controlling channels on AIX, Linux, and Windows” on page 264](#)
 -  [“Monitoring and controlling channels on IBM i” on page 287](#)

Related concepts

[“Setting up IBM MQ for z/OS” on page 957](#)

Use this topic as a step-by-step guide for customizing your IBM MQ for z/OS system.

Related tasks

[“Configuring connections between the client and server” on page 16](#)

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.

[“Configuring a queue manager cluster” on page 309](#)

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.

[“Changing IBM MQ configuration information in .ini files on Multiplatforms” on page 94](#)

You can change the behavior of IBM MQ or an individual queue manager to suit the needs of your installation by editing the information in the configuration (.ini) files. You can also change configuration options for IBM MQ MQI clients.

[“Configuring queue managers on z/OS” on page 952](#)

Use these instructions to configure queue managers on IBM MQ for z/OS.

[“Setting up communications with other queue managers on z/OS” on page 1030](#)

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

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 AIX, Linux, and Windows](#)
-  [Message channel planning example for IBM i](#)
-  [Message channel planning example for z/OS](#)
-  [Message channel planning example for z/OS using queue sharing groups](#)

Related concepts

[Channels](#)

[Introduction to message queuing](#)

[Distributed queuing and clusters](#)

Related tasks

[“Configuring distributed queuing” on page 212](#)

This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

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 212](#). If your queue manager is in a cluster, message flow is controlled using different techniques, as described in [“Message flow control” on page 213](#).  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 214](#)

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 214](#)

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 17 on page 215](#) 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.

<i>Table 17. Three ways of using the remote queue definition object</i>			
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 216](#) 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.

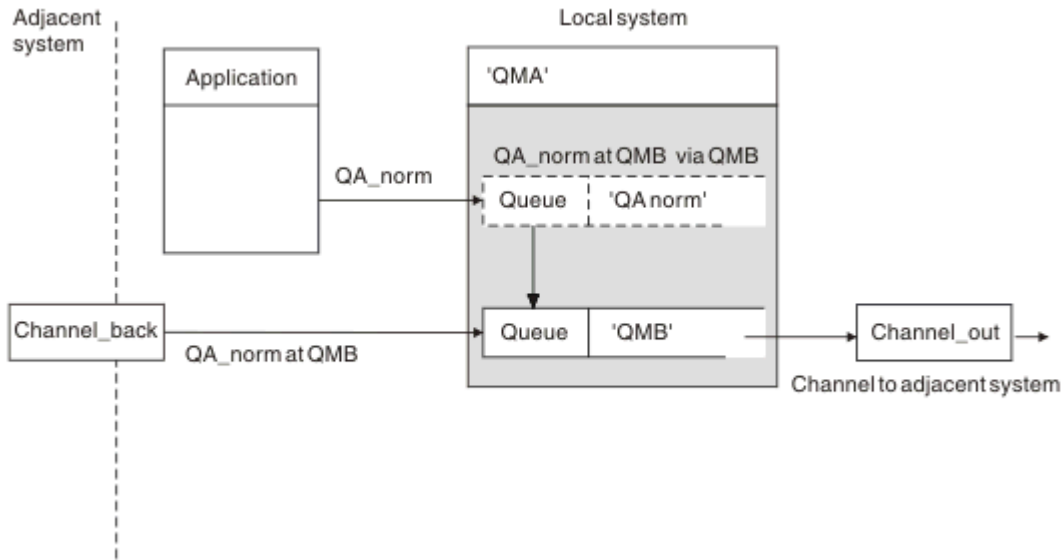


Figure 6. A remote queue definition is used to resolve a queue name to a transmission queue to an adjacent queue manager

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.

Related reference

[Queue name resolution](#)

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.

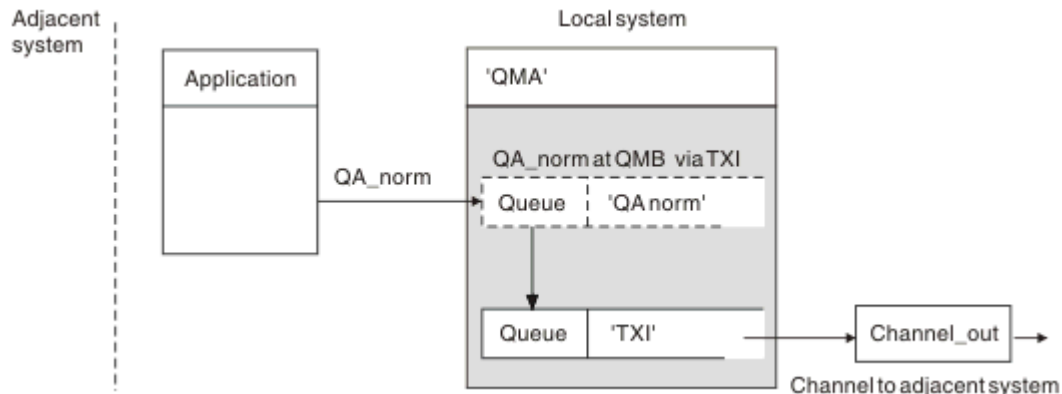


Figure 7. The remote queue definition allows a different transmission queue to be used

In a distributed-queueing 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 216 in [“Putting messages on remote queues”](#) on page 215. Figure 7 on page 217 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 217, you must provide the remote queue object 'QA_norm', and the transmission queue 'TXI'. You must provide 'QA_norm' to choose the 'QA_norm' queue at the remote queue manager, the transmission queue 'TXI', and the queue manager 'QMB_priority'. Specify 'TXI' in the definition of the channel adjacent to the system.

Messages are placed on transmission queue 'TXI' 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 309.

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.

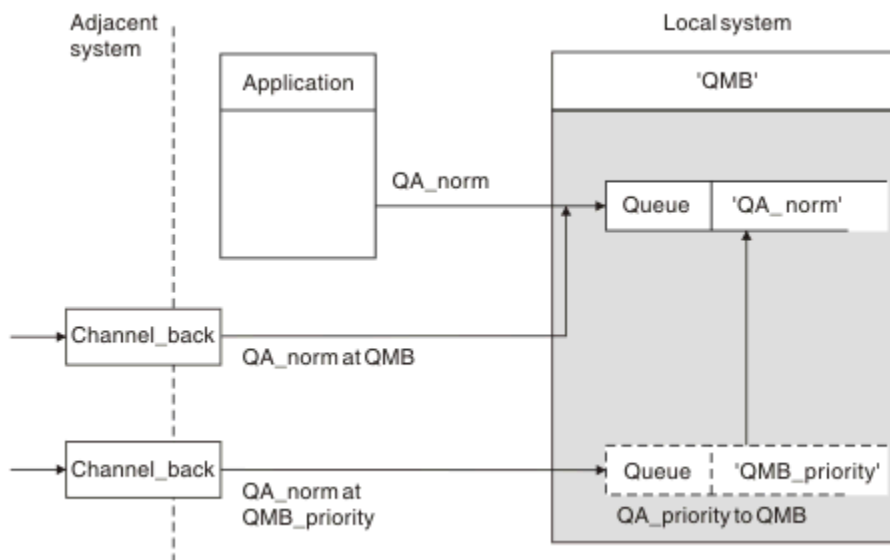


Figure 8. Receiving messages directly, and resolving alias queue manager name

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 218](#) 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.

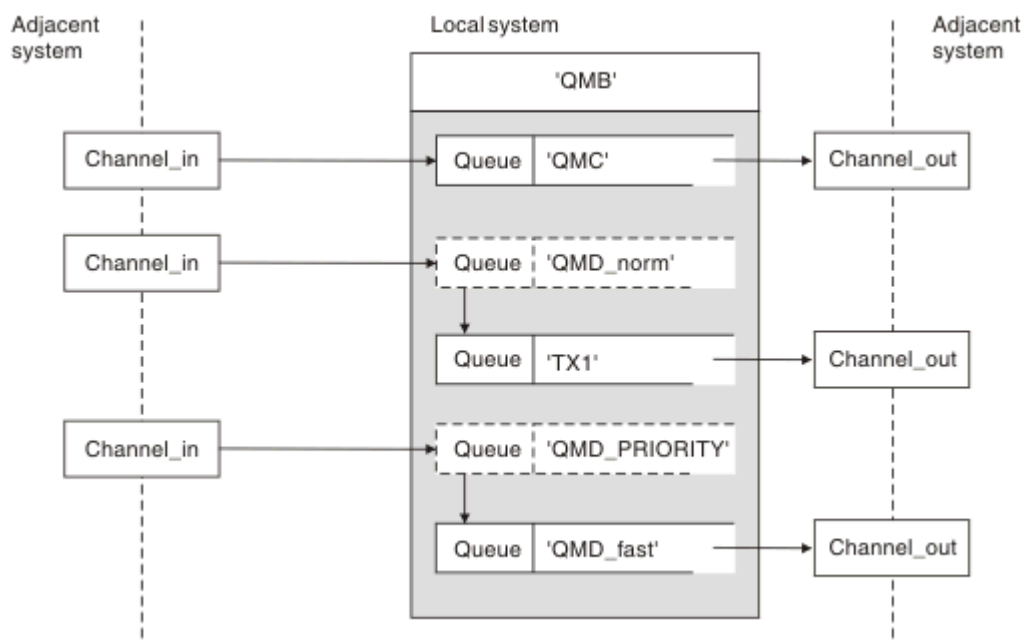


Figure 9. Three methods of passing messages through your system

The technique shown in [Figure 8 on page 218](#) in “[Receiving messages](#)” on page 218, showed how an alias flow is captured. [Figure 9 on page 219](#) 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 set up 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.

Reasons for separating messages into different message flows

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.

Example message flow

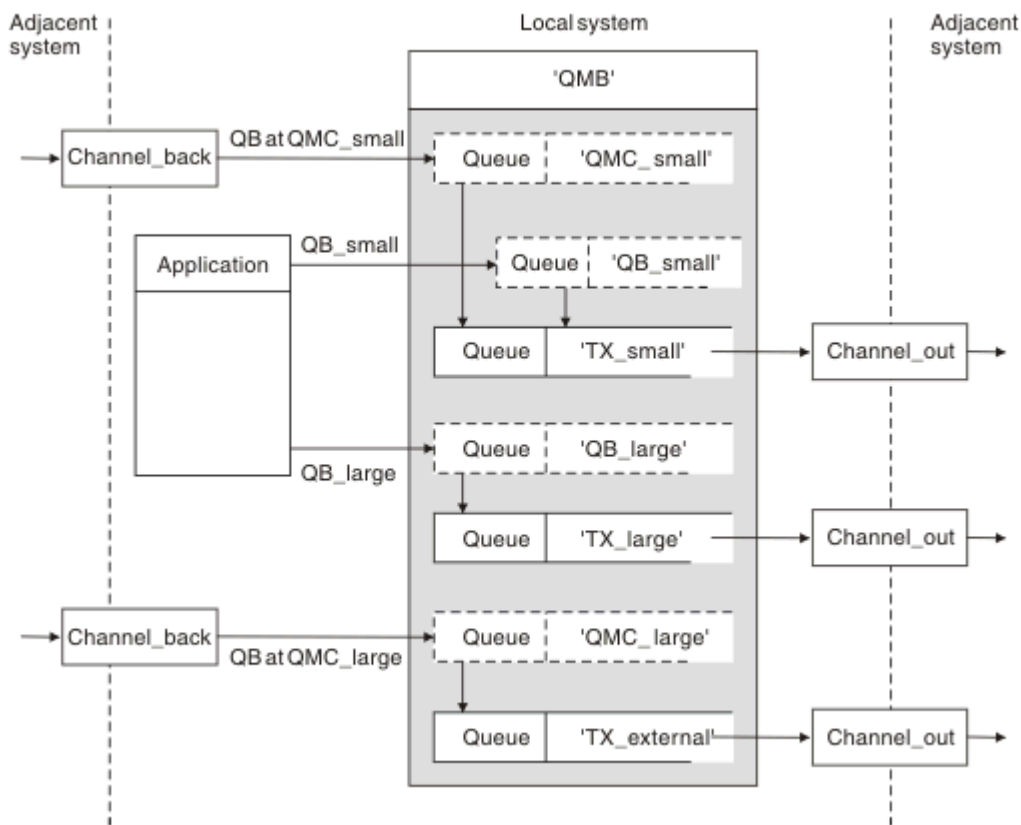


Figure 10. Separating messages flows

In the example shown in [Figure 10 on page 221](#), 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.

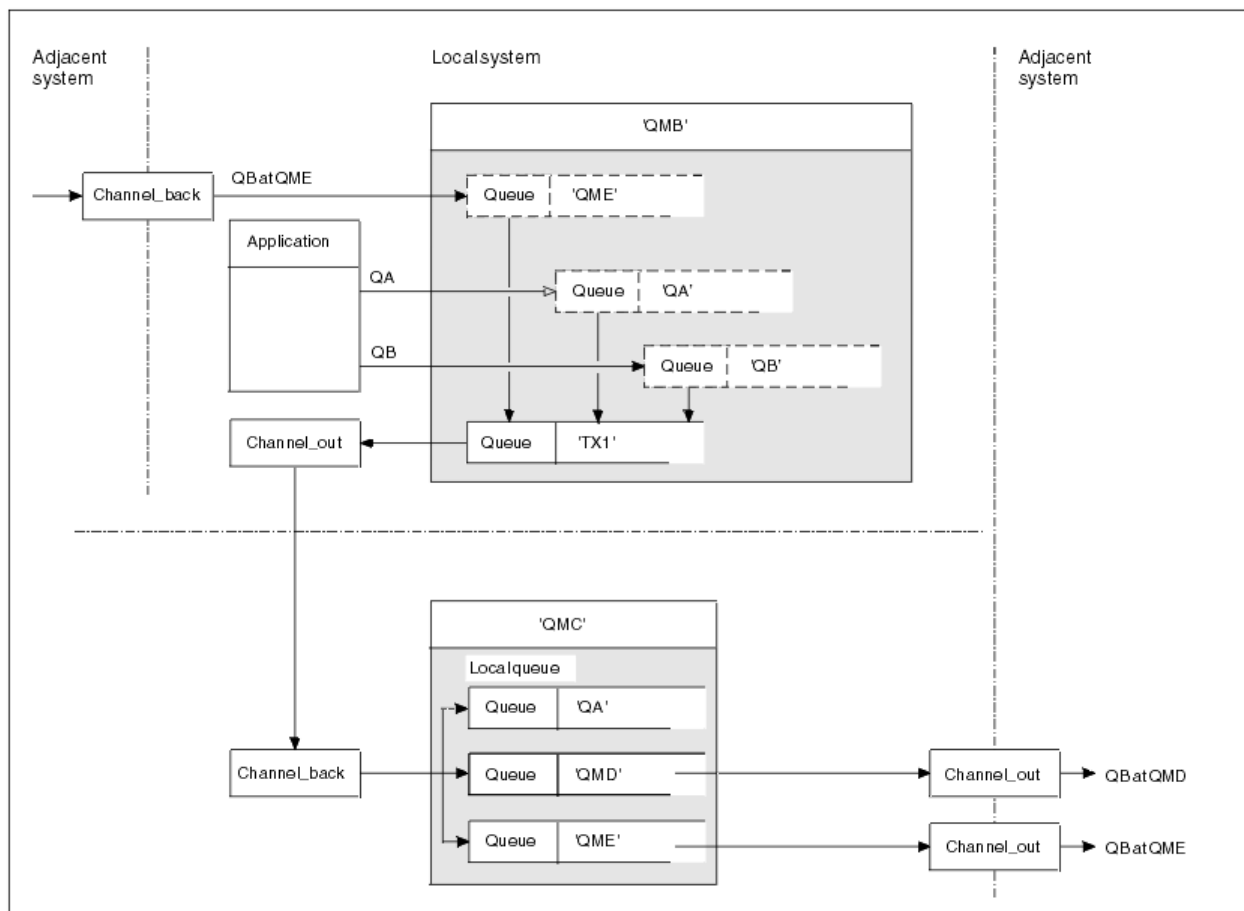


Figure 11. Combining message flows on to a channel

Figure 11 on page 222 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.

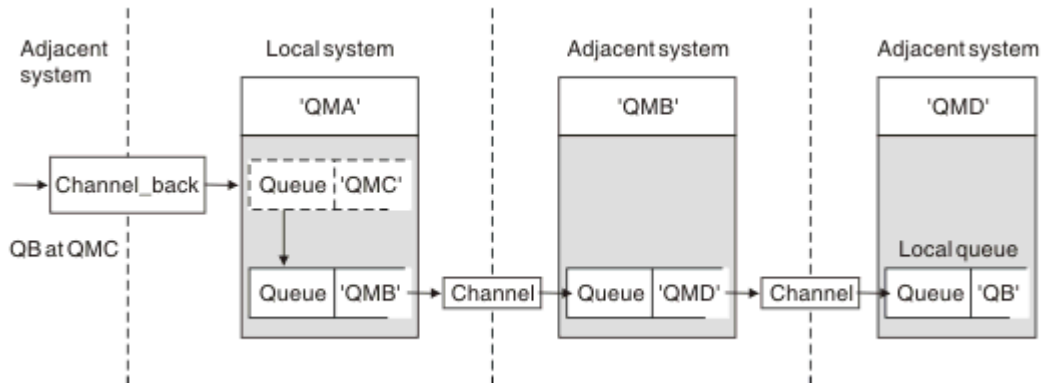


Figure 12. Diverting message streams to another destination

Figure 12 on page 223 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 400.

Sending messages to a distribution list

On Multiplatforms, you can use a single MQPUT call to have an application send a message to several destinations.

In IBM MQ on Multiplatforms, 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.

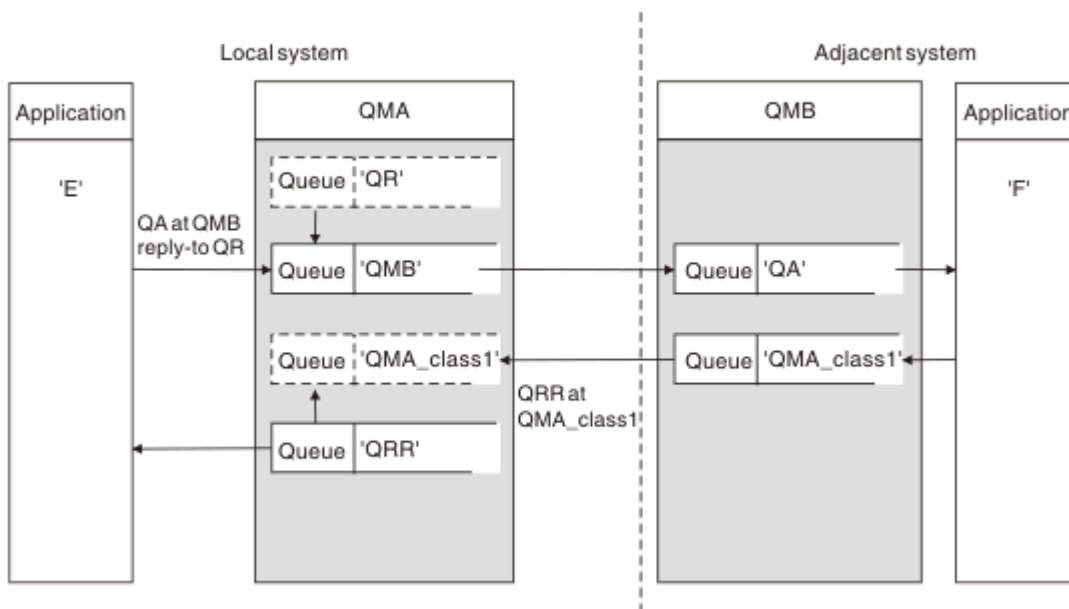


Figure 13. Reply-to queue name substitution during PUT call

A complete remote queue processing loop using a reply-to queue is shown in [Figure 13](#) on page 224. This loop applies in both a distributed-queuing environment and a clustering environment. The details are as shown in [Table 21](#) on page 231.

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 separated 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 214](#)

This topic explains the three ways that you can create a remote queue definition.

[“Reply-to queue alias example” on page 226](#)

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 227](#)

An explanation of the example and how the queue manager uses the reply-to queue alias.

[“Reply-to queue alias walk-through” on page 228](#)

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 226, the return route must be available for the reply messages, including the transmission queue, channel, and queue manager alias.

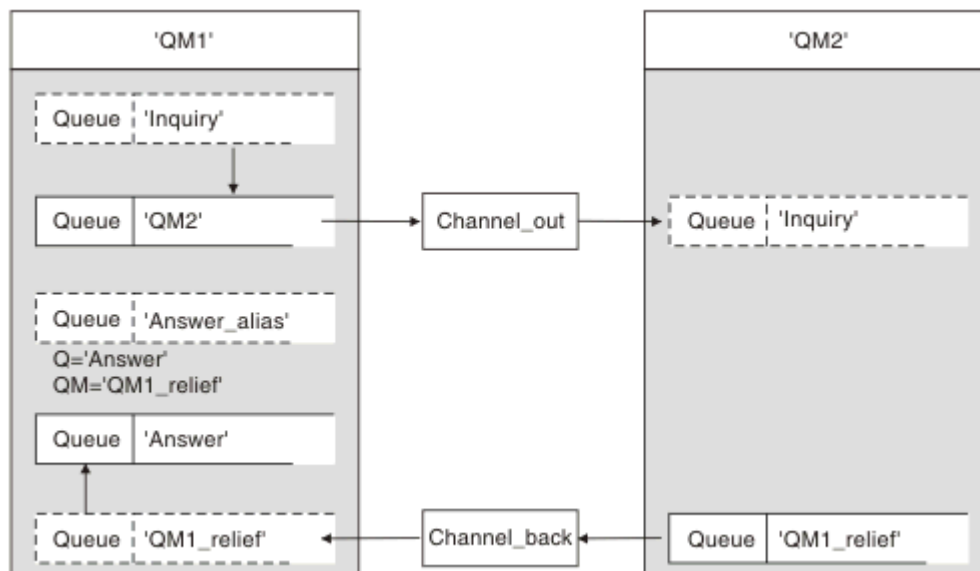


Figure 14. Reply-to queue alias example

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 18. 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

Reply-to queue manager name

Answer_alias

(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

Reply-to queue manager name

Answer

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](#).

 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 19 on page 231](#), [Table 20 on page 231](#), and [Table 21 on page 231](#).

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 19. 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 20. 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 21. 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 number mismatches are first detected only when messages begin to flow across the channel. 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.

Resetting or ignoring sequence number mismatches, see [**IgnoreSeqNumberMismatch**](#) in *Channels stanza of the qm.ini file*, does not risk losing or duplicating a batch of messages, and it does not reset the in-doubt status of a channel.

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 Multiplatforms 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. For more information, see [Message route techniques](#).

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 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:

-  [“Monitoring and controlling channels on AIX, Linux, and Windows” on page 264](#)
-  [“Monitoring and controlling channels on z/OS” on page 1033](#)
-  [“Monitoring and controlling channels on IBM i” on page 287](#)

For message channel planning examples, see:

-  [Message channel planning example for AIX, Linux, and Windows](#)
-  [Message channel planning example for IBM i](#)
-  [Message channel planning example for 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 235](#)

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 242](#)

The channel control function provides facilities for you to define, monitor, and control channels.

[“What happens when a message cannot be delivered?” on page 256](#)

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 260](#)

The handling of channel initialization data depends on your IBM MQ platform.

[“Data conversion for messages” on page 262](#)

IBM MQ messages might require data conversion when sent between queues on different queue managers.

[“Writing your own message channel agents” on page 262](#)

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 263](#)

Other topics to consider when preparing IBM MQ for distributed queue management. This topic covers undelivered-message queues, queues in use, system extensions and user-exit programs, and running channels and listeners as trusted applications.

Related reference

[Example configuration information](#)

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.



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 MOI.

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 ( 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.

-  On z/OS, you can also use the Operation and Control panels described in [Administering IBM MQ for z/OS](#).
-  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

[“IBM MQ distributed queuing techniques” on page 213](#)

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 233](#)

Distributed queue management (DQM) is used to define and control communication between queue managers.

[“Triggering channels” on page 257](#)

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 255](#)

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.

Related tasks

[“Creating queue managers on Multiplatforms” on page 7](#)

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

[“Monitoring and controlling channels on AIX, Linux, and Windows” on page 264](#)

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 287](#)

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 16](#)

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.

[“Configuring a queue manager cluster” on page 309](#)

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 communications with other queue managers on z/OS” on page 1030](#)

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

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 Multiplatforms, you do not have to specify the protocol. 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 Multiplatforms, you do not have to specify the protocol. 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

 This is required. You must use the initiation queue called SYSTEM.CHANNEL.INITQ.

 This is optional. Consider naming the initiation queue SYSTEM.CHANNEL.INITQ.

- Transmission queue definition

A local queue with the USAGE attribute set to XMITQ.  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 239](#)

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 239](#)

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 240](#).

```

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 240](#).

```

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 241](#).

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          1-99999999, *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 AIX, Linux, and Windows” on page 264.](#)

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

Related concepts

[Where to find information to help with problem determination](#)

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 Multiplatforms, 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 212.

For information about defining MQI channels, see [“Defining MQI channels”](#) on page 30.

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 [“Handling in-doubt channels”](#) on page 254.

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 244 shows the hierarchy of all possible channel states and the substates that apply to each of the channel states.

Figure 21 on page 245 shows the links between channel states. These links apply to all types of message channel and server-connection channels.

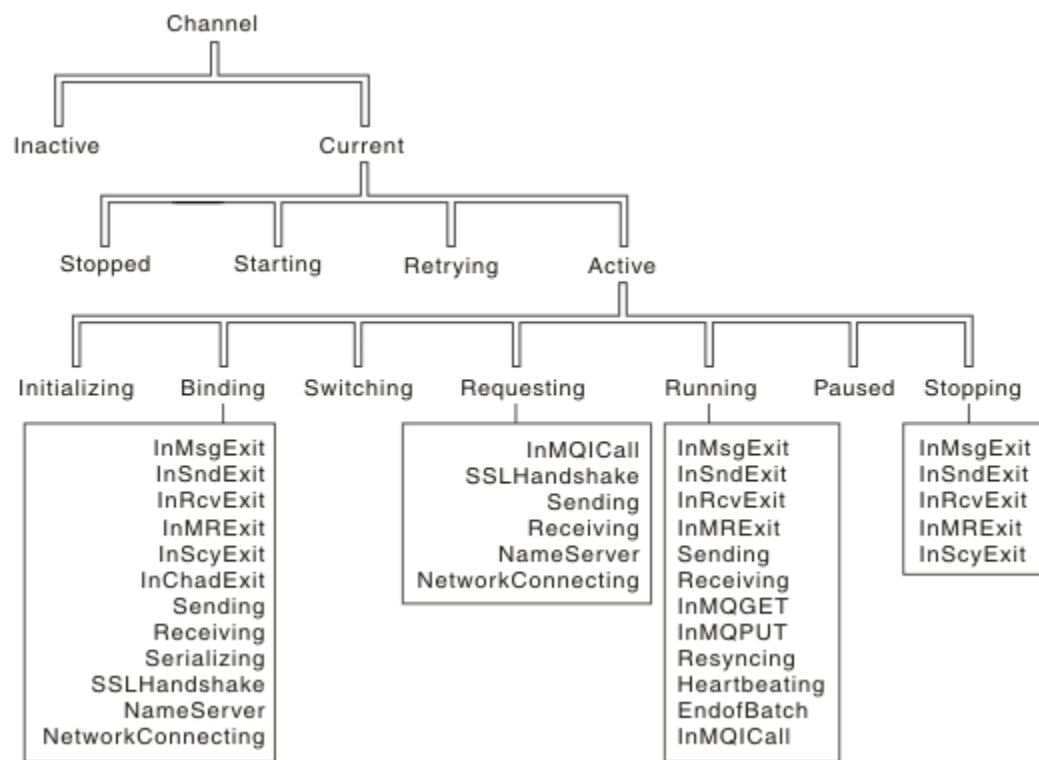


Figure 20. Channel states and substates

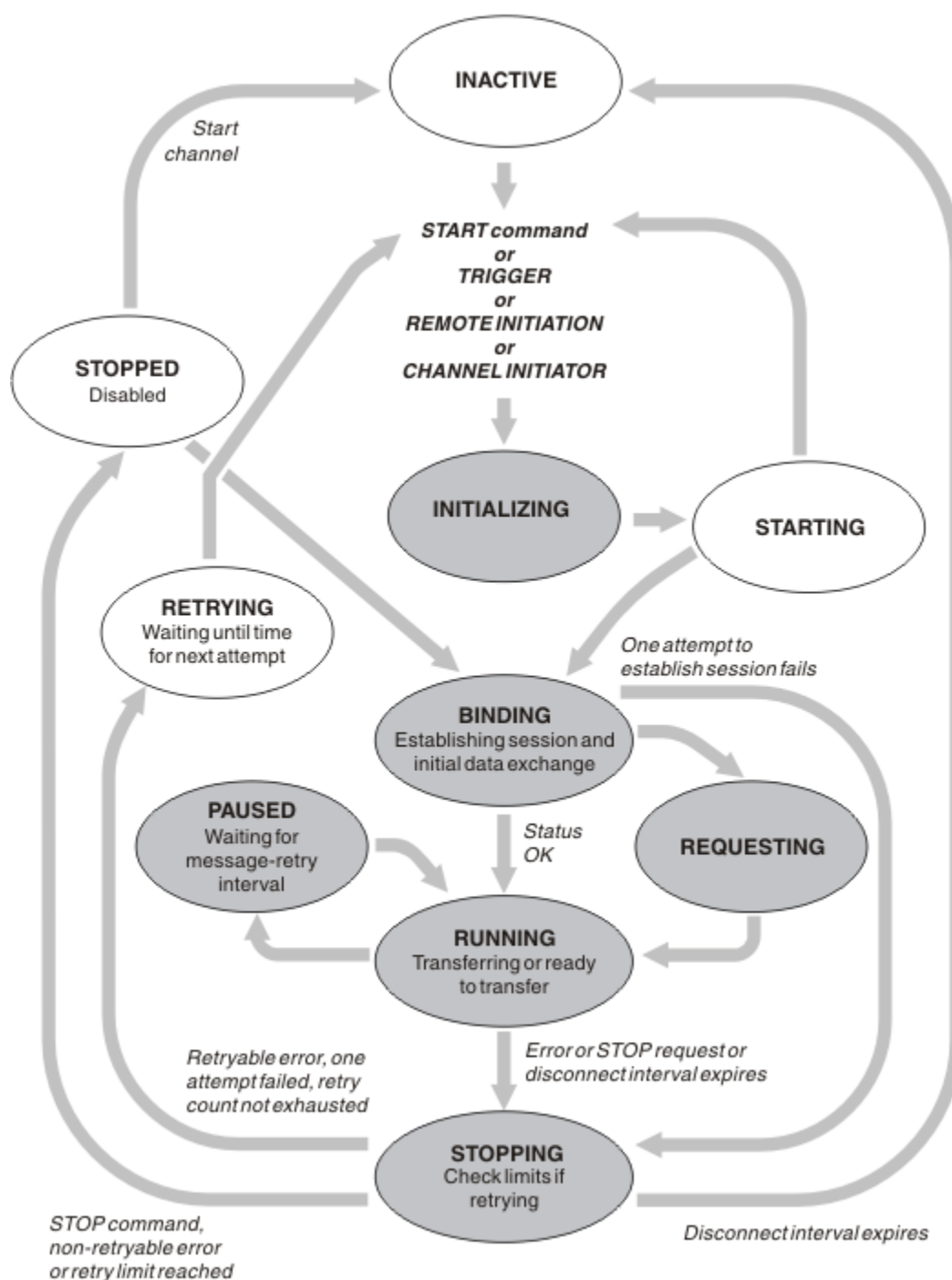


Figure 21. Flows between channel states

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 245](#).

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 244](#).

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.

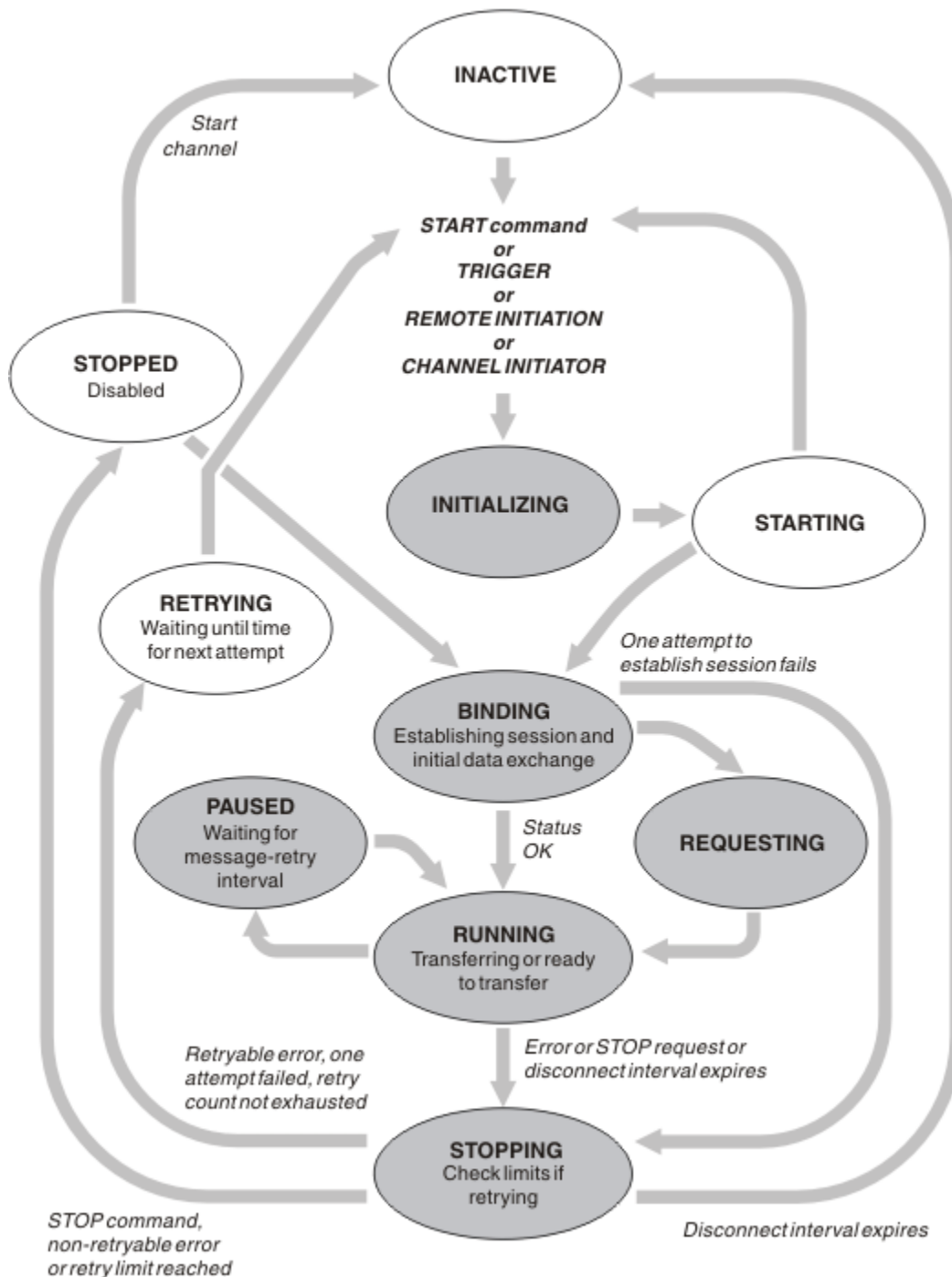


Figure 22. Flows between channel states

Note:

1. When a channel is in one of the six states highlighted in Figure 22 on page 246 (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 for your platform:

-  Use the ALTER QMGR MAXCHL command.
-  Edit the queue manager initialization file.
-   Edit the queue manager configuration file.
- Use the IBM 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 the following topics:

-  [Administering IBM MQ](#).
-  [Administering IBM MQ for IBM i](#).
-  [Administering 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  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 the following topics:

-  [Administering IBM MQ.](#)
-  [Administering IBM MQ for IBM i.](#)
-  [Administering 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 253).

Note: For  IBM i, AIX, 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.

 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.

 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 with the **MAXINST** parameter, and to prevent a single client application from exhausting server-connection channel capacity with the **MAXINSTC** parameter.

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.

Performance considerations

The queue manager must perform additional processing when accepting new client connections if the number of instances of the server-connection channel is greater than its **MAXINSTC** value but less than its **MAXINST** value. If the queue manager has a large total number of active channels, you might see additional computation overhead and latency when establishing new client connections to channels with **MAXINSTC** less than **MAXINST**.

The use of **MAXINSTC** is not recommended for queue managers with over 1000 concurrent channel instances, in particular if large numbers of applications create short-lived connections concurrently, as this pattern increases competition for serialized resources necessary to validate the **MAXINSTC** count.

Avoid high rates of connections over such server-connection channels by increasing their **SHARECNV** value to encourage sharing. Also, consider splitting clients across multiple server-connection channel definitions with only **MAXINST** configured to limit the number of channel instances.

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

z/OS On 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 lower 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 \(KAINT\)](#).

Multi On IBM i, AIX, 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. If 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, AIX, 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**.

z/OS On 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 following values, depending on the **RCVTTYPE** parameter, and subject to any limit imposed by **RCVTMIN** if it applies:
 - The negotiated **HBINT** multiplied by a constant
 - The negotiated **HBINT** plus a constant number of seconds
 - A constant number of seconds

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. 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.

Suggested settings

IBM MQ for z/OS

As an initial starting point, you can use:

```
/cpf ALTER QMGR TCPKEEP(YES) RCVTTYPE(ADD) RCVTIME(60) ADOPTMCA(ALL) ADOPTCHK(ALL)
```

where `cpf` is the command prefix for the queue manager subsystem.

These parameters help detect when the other end of the channel is no longer available. Without the recommended the **TCPKEEP** and **RCVTIME** timeout parameters, a sender or receiver channel can remain in STATUS(RUNNING) with zero messages moving across the channel. Manual intervention is required to stop this frozen channel so that it can be restarted.

Note: For z/OS queue managers, changes to parameters like **TCPKEEP**, **RCVTTYPE**, and **RCVTIME** take effect only for channels that are later started. The channels that are currently running remain unaffected. To apply these changes to long-running channels, you must restart the channels or recycle the CHIN.

See [ALTER QMGR](#) and [IBM MQ network availability](#) for more information on the various parameters.

If the IP address of the sender could translate to more than one address, you might need to set **ADOPTCHK** to QMNAME rather than ALL.

For z/OS queue managers, based on the description of the parameters for **ALTER QMGR**, the changes to those particular parameters are dynamic. They take effect for channels that are started later. If you want them to take effect for long-running channels, you either need to restart the channels or recycle the **CHIN** since there are many channels.

IBM MQ for Multiplatforms

In `qm.ini`, add the following information:

```
TCP:
KeepAlive=Yes
CHANNELS:
AdoptNewMCA=ALL
AdoptNewMCACheck=ALL
```

See [ALTER QMGR](#), [Configuration file stanzas for distributed queuing](#), and [“Channels stanza of the qm.ini file”](#) on page 127 for more information.

If the IP address of the sender could translate to more than one address, you might need to set **AdoptNewMCACheck** to QMNAME rather than **ALL**.

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.

Related tasks

[Administering IBM MQ](#)

 [Administering IBM MQ for z/OS](#)

 [Administering IBM MQ for 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
-   other platform-specific mechanisms, as follows:

 **For z/OS:**
The Stop a channel panel

 **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.

 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.

 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.

About this task

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.

 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.

 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.

Procedure

- Restart the channel in one of the following ways:

- By using the [START CHANNEL MQSC command](#).
- By using the [Start Channel PCF command](#).
- By using the [IBM MQ Explorer](#)
-  On z/OS, by using the [Start a channel panel](#).
-  On IBM i, by using either the [STRMQMCHL CL command](#) or the START option on the [WRKMQMCHL panel](#).

Handling 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.

About this task

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.

Procedure

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.

Note: 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 i, the **DISPLAY CHSTATUS** command can be executed from a file using the **STRMQMQSC** command or the Work with MQM Channel Status CL command, **WRKMQMCHST**.

2. If the two LUWIDs are the same, use the **RESOLVE CHANNEL** command to commit the in-doubt messages.

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 **RESOLVE CHANNEL** command:

```
RESOLVE CHANNEL(name) ACTION(COMMIT)
```

3. If the two LUWIDs are different, use the **RESOLVE CHANNEL** command to back out the in-doubt messages.

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 **RESOLVE CHANNEL** command:

```
RESOLVE CHANNEL( name ) ACTION(BACKOUT)
```

 On IBM i, you can use the Resolve MQM Channel command, **RSVMQMCHL**.

Results

Once this process is complete the channel is no longer in doubt. The transmission queue can now be used by another channel, if required.

Related reference

[DISPLAY CHSTATUS \(display channel status\)](#)

[RESOLVE CHANNEL \(ask a channel to resolve in-doubt messages\)](#)

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 256.

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 256 shows the processing that occurs when an MCA is unable to put a message to the destination queue. (The options shown do not apply to all platforms.)

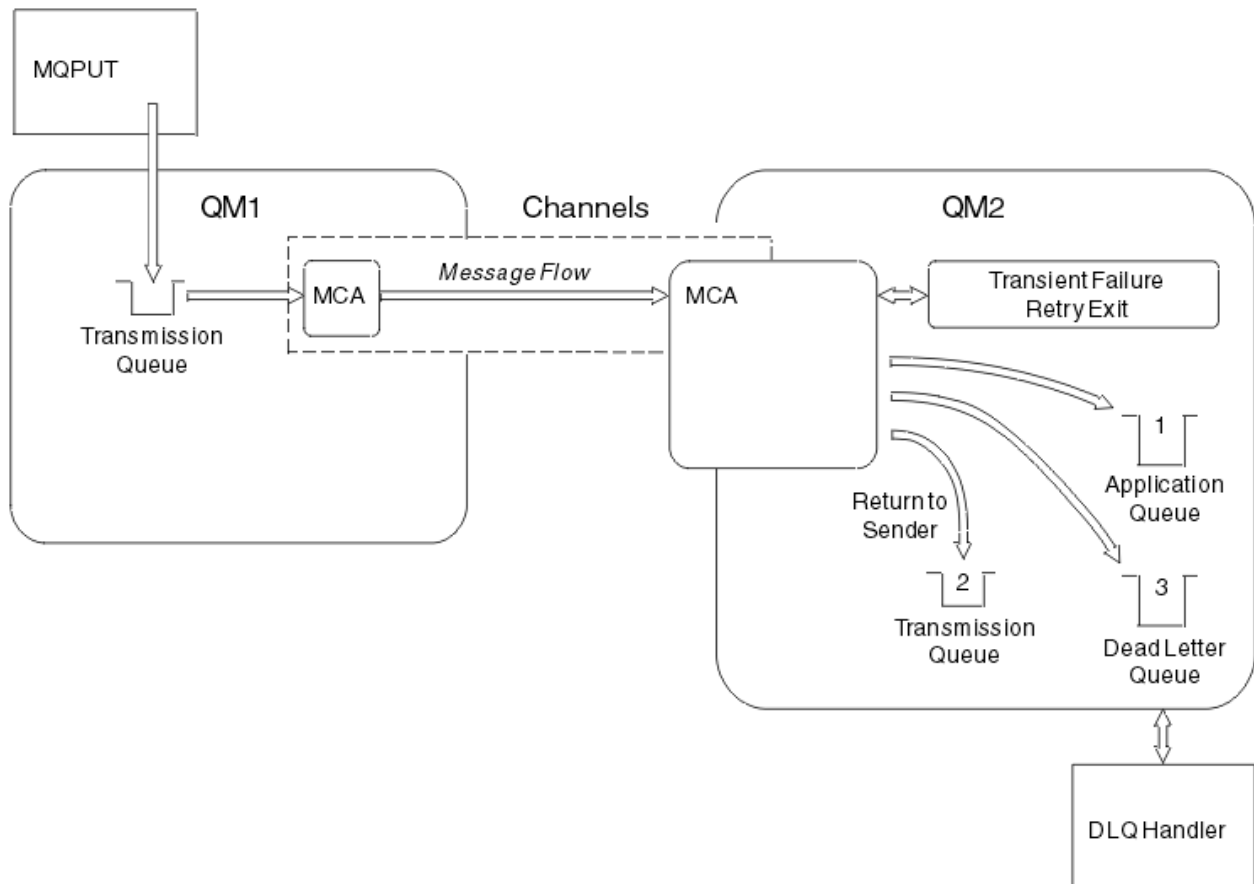


Figure 23. What happens when a message cannot be delivered

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 here:

-  [Processing messages on a dead-letter queue](#)
-  [The dead-letter queue handler utility \(CSQUDLQH\)](#)

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 AIX, Linux, and Windows, see [“Triggering channels on AIX, Linux, and Windows.” on page 259](#)
-  For IBM i, see [“Triggering channels in IBM MQ for IBM i” on page 259](#)
-  For z/OS, see [“Transmission queues and triggering channels” on page 1032](#)

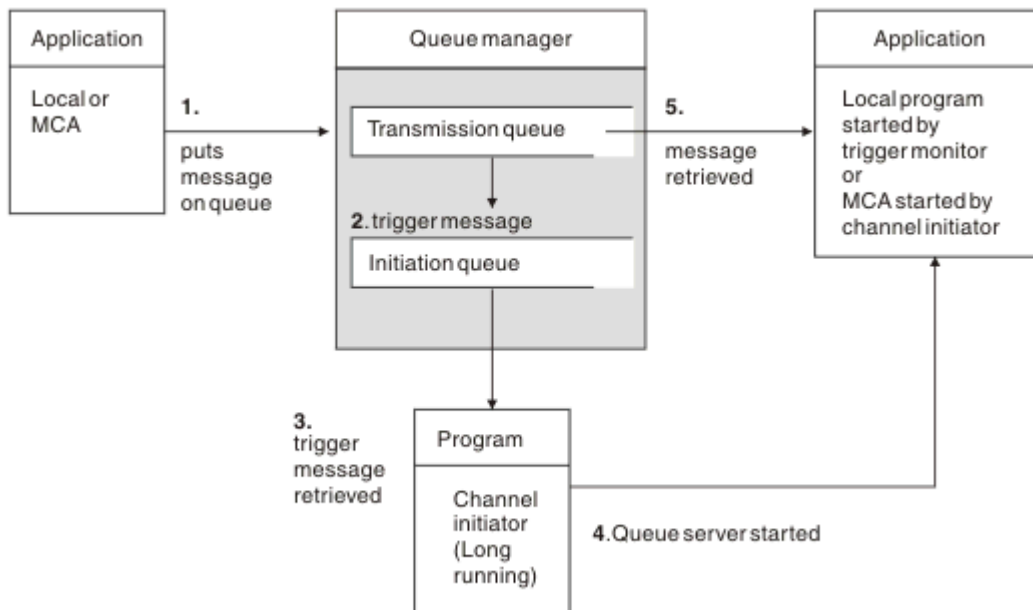


Figure 24. The concepts of triggering

The objects required for triggering are shown in [Figure 24 on page 258](#). 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.  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, AIX, 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 AIX, Linux, and Windows.

ALW

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)
```

Deprecated 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 260](#)

Triggering is implemented using the channel initiator process.

Related tasks

[“Configuring distributed queuing” on page 212](#)

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 AIX, Linux, and Windows](#)

 [Intercommunication jobs on 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 MQ for Multiplatforms 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 AIX 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.

IBM MQ for 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 more information about CSQINP2 and CSQINPX, see [Customize the initialization input data sets](#), and [ALTER QMGR](#).

IBM MQ for Multiplatforms



In IBM MQ for Multiplatforms, 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 described in [“IBM MQ configuration file, mqs.ini” on page 96](#).

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 AIX and Linux systems:

```
/var/mqm/qmgrs/QMNAME/qm.ini
```

For Windows systems:

```
C:\ProgramData\IBM\MQ\qmgrs\QMNAME\qm.ini
```



For IBM i:

```
/QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini
```

Here is an extract from a `qm.ini`. It specifies that the TCP/IP listener is to listen on port 2500, the maximum number of current channels is 200, and the maximum number of active channels is 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.

For more information about `qm.ini` files, see [“Queue manager configuration files, qm.ini” on page 108](#).

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 queues, 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 AIX and Linux systems; for more information, see [The sample DLQ handler, amqsdldq](#).

 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

 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 AIX, Linux, and Windows

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.

About this task

You can use the following types of command to control channels:

The IBM MQ commands (MQSC)

You can use the MQSC as single commands in an MQSC session in AIX, Linux, and Windows 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 [Administering IBM MQ for Multiplatforms using control commands](#).

Programmable command format commands

For details, see [PCF commands](#).

IBM MQ Explorer

On 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 22 on page 266](#).

Procedure

- [“Functions required for setting up and controlling channels” on page 265](#)
- [“Getting started with objects” on page 267](#)
- [“Setting up communication on Windows” on page 274](#)
- [“Setting up communication on AIX and Linux” on page 281](#)

Related tasks

[“Monitoring and controlling channels on IBM i” on page 287](#)

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 AIX, Linux, and Windows](#)

 [Message channel planning example for AIX, Linux, and Windows](#)

[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:

- [“Getting started with objects” on page 267](#)
- [“Creating associated objects” on page 268](#)
- [“Creating default objects” on page 268](#)
- [“Creating a channel” on page 268](#)
- [“Displaying a channel” on page 269](#)
- [“Displaying channel status” on page 269](#)

- [“Checking links using Ping” on page 270](#)
- [“Starting a channel” on page 270](#)
- [“Stopping a channel” on page 272](#)
- [“Renaming a channel” on page 272](#)
- [“Resetting a channel” on page 273](#)
- [“Resolving in-doubt messages on a channel” on page 273](#)

Table 22 on page 266 shows the full list of IBM MQ functions that you might need.

Table 22. Functions required in AIX, Linux, and Windows 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	dlmqm		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
Delete queue		DELETE QALIAS DELETE QLOCAL DELETE QMODEL DELETE QREMOTE See, DELETE queues .	Yes

Table 22. Functions required in AIX, Linux, and Windows systems (continued)			
Function	Control commands	MQSC	IBM MQ Explorer equivalent?
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, on the following platforms only:  AIX  Windows Windows systems		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 AIX, Linux, and Windows](#).

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 [Administering IBM MQ for Multiplatforms using 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 AIX, Linux, and Windows. 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 AIX and Linux, use Ctrl-d.
- Alternatively, on AIX, Linux, and Windows, 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.

Related tasks

[Using Ping to test communications](#)

[Pinging a channel to verify a connection](#)

Related reference

[PING CHANNEL \(test channel response\)](#)

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 AIX and Linux, inetd can be configured to start a channel. inetd is started as a separate process.
-   For LU 6.2 in AIX and Linux, configure your SNA product to start the LU 6.2 responder process.
-  For LU 6.2 in Windows, 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_DATA_PATH\qmgrs\qmname\errors\AMQERR01.LOG (for each queue manager called qmname)

MQ_DATA_PATH\qmgrs\@SYSTEM\errors\AMQERR01.LOG (for general errors)

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

Note: On Windows, you still also get a message in the Windows systems application event log.

AIX and Linux

/var/mqm/qmgrs/qmname/errors/AMQERR01.LOG (for each queue manager called qmname)

/var/mqm/qmgrs/@SYSTEM/errors/AMQERR01.LOG (for general errors)

On AIX, Linux, and Windows, use the **runmqclsr** 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.

```
runmqclsr -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 253](#).

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 267](#)

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 242](#)

The channel control function provides facilities for you to define, monitor, and control channels.

Related tasks

[“Configuring distributed queuing” on page 212](#)

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 267](#)

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 242](#)

The channel control function provides facilities for you to define, monitor, and control channels.

Related tasks

[“Configuring distributed queuing” on page 212](#)

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 on Windows

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, the connection must be defined and available. This section explains how to do this by using the forms of communication available for IBM MQ for Windows systems.

Before you begin

You might find it helpful to refer to [Example configuration - IBM MQ for Windows](#).

  A message channel that uses TCP/IP can be pointed at an IBM Aspera faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. A queue manager running on any entitled platform can connect through an Aspera gateway. The gateway itself is deployed on Red Hat or Ubuntu Linux, or Windows. See [Defining an Aspera gateway connection on Linux or Windows](#).

About this task

When setting up communication for IBM MQ on Windows, you can choose from the following types of communication:

- TCP/IP
- LU 6.2
- NetBIOS

Procedure

- For information on setting up communication for your Windows system, see the subtopic for your chosen communication type:
 - [“Defining a TCP connection on Windows” on page 275](#)
 - [“Defining an LU 6.2 connection on Windows” on page 277](#)
 - [“Defining a NetBIOS connection on Windows” on page 278](#)

Not all of the functions and facilities of IBM MQ for Windows are available in environments that use communications protocols other than TCP/IP. The item that is not available is IBM MQ Explorer.

Related tasks

[“Monitoring and controlling channels on AIX, Linux, and Windows” on page 264](#)

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 16](#)

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 communication on AIX and Linux” on page 281](#)

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, the connection must be defined and available. This section

explains how to do this by using the forms of communication available for IBM MQ for UNIX or Linux systems.

Related reference

“Which communication type to use” on page 17

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

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.

Before you begin

  A message channel that uses TCP/IP can be pointed at an IBM Aspera faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. A queue manager running on any entitled platform can connect through an Aspera gateway. The gateway itself is deployed on Red Hat or Ubuntu Linux, or Windows. See [Defining an Aspera gateway connection on Linux or Windows](#).

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('OS2R0G3(1822)') +  
    XMITQ('XMITQ name') +  
    REPLACE
```

where OS2R0G3 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 [runmqslr](#) command.

A basic example of using the **runmqslr** command:

```
runmqslr -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 263. 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 249.

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 IBM MQ for Multiplatforms”](#) on page 284 for more information, and the specific value for Windows.

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 23. 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.	-
AIX and Linux systems	The same as in the corresponding side information about the remote queue manager.	<code>MQ_INSTALLATION_PATH/bin/amqcrs6a</code>
Windows	As specified in the Windows Run Listener command, or the invokable Transaction Program that was defined using TpSetup on Windows.	<code>MQ_INSTALLATION_PATH\bin\amqcrs6a</code>

`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 Windows” on page 277](#)

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 Windows” on page 278](#)

Receiving channel programs are started in response to a startup request from the sending channel.

Windows Sending end on LU 6.2 on Windows

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	OS2R0G2
Partner TP Name	recv
Mode Name	#INTER

Receiving on LU 6.2 on Windows

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 **-m** parameter used in the last part of this command 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

A NetBIOS connection applies only to a client and server running 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 requires 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 279](#)

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 280](#)

The queue manager limits for NetBIOS sessions, commands, and names can be specified in two ways.

[“Establishing the LAN adapter number” on page 280](#)

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 runmqslsr command.

[“Initiating the NetBIOS connection” on page 280](#)

Defining the steps needed to initiate a connection.

[“Defining the target listener for the NetBIOS connection” on page 281](#)

Defining the steps to be undertaken at the receiving end of the NetBIOS connection.

Windows

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 **runmqslsr** command, for example:

```
runmqslsr -t netbios -l my_station
```

2. The **MQNAME** environment variable with a value that is established by the command:

```
SET MQNAME= my_station
```

For example:

```
SET MQNAME=CLIENT1
```

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.

Windows 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
```

Windows 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 runmqslsr 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.

Windows 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) +  
CONNNAME(your_station) +  
XMITQ(xmitq) +  
MCATYPE(THREAD) +  
REPLACE
```

Windows Defining the 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 263. 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.

Linux AIX Setting up communication on AIX and Linux

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, the connection must be defined and available. This section explains how to do this by using the forms of communication available for IBM MQ for UNIX or Linux systems.

Before you begin

You might find it helpful to refer to the following sections:

- [AIX](#) [Example configuration - IBM MQ for AIX](#)
- [Linux](#) [Example configuration - IBM MQ for Linux](#)

[MQ Adv.](#) [CD](#) A message channel that uses TCP/IP can be pointed at an IBM Aspera faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. A queue manager running on any entitled platform can connect through an Aspera gateway. The gateway itself is deployed on Red Hat or Ubuntu Linux, or Windows. See [Defining an Aspera gateway connection on Linux or Windows](#).

About this task

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.

When setting up communication for IBM MQ on AIX and Linux, you can choose from the following types of communication:

- TCP/IP
- LU 6.2

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. See [IBM MQ MQI clients](#).

Procedure

For information on setting up communication for your AIX or Linux system, see the subtopic for your chosen communication type:

- [“Defining a TCP connection on AIX and Linux” on page 282](#)
- [“Defining an LU 6.2 connection on AIX and Linux” on page 286](#)
-   [“Defining an Aspera gateway connection on Linux or Windows platforms” on page 942](#)

Related tasks

[“Monitoring and controlling channels on AIX, Linux, and Windows” on page 264](#)

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 16](#)

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 communication on Windows” on page 274](#)

When a distributed-queuing management channel is started, it tries to use the connection specified in the channel definition. For this to succeed, the connection must be defined and available. This section explains how to do this by using the forms of communication available for IBM MQ for Windows systems.

Related reference

[“Which communication type to use” on page 17](#)

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

Defining a TCP connection on AIX 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.

Before you begin

  A message channel that uses TCP/IP can be pointed at an IBM Aspera faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. A queue manager running on any entitled platform can connect through an Aspera gateway. The gateway itself is deployed on Red Hat or Ubuntu Linux, or Windows. See [Defining an Aspera gateway connection on Linux or Windows](#).

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 use the extended inet daemon (xinetd) instead of the inet daemon. For more information on how to use the extended inet daemon on a Linux system, see [Step 2 of Example: setting up IBM MQ cross-platform communication on Linux](#).

Related concepts

[“Using the TCP/IP listener on AIX and Linux” on page 283](#)

To start channels on AIX and Linux, the `/etc/services` file and the `inetd.conf` file must be edited

[“Using the TCP listener backlog option on IBM MQ for Multiplatforms” on page 284](#)

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 285](#)

To run the listener supplied with IBM MQ, which starts new channels as threads, use the `runmq1sr` command.

[“Using the TCP/IP SO_KEEPALIVE option” on page 285](#)

On some AIX 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 on AIX and Linux

To start channels on AIX 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 Linux systems:

```
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 IBM MQ for Multiplatforms”](#) on page 284.

Using the TCP listener backlog option on IBM MQ for Multiplatforms

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 24 on page 284](#).

Table 24. Maximum outstanding connection requests queued at a TCP/IP port	
Server platform	Maximum connection requests
 AIX	100
 Linux	100
 IBM i	255
 Windows Server	100

If the backlog reaches the values shown in [Table 24 on page 284](#), 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 24 on page 284](#)) 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 enabled either:

- Use the `runmqclsr -b` command, or
- Use the MQSC command **DEFINE LISTENER** with the `BACKLOG` attribute set to the required value.

For information about the `runmqclsr` command, see [runmqclsr](#). For information about the `DEFINE LISTENER` command, see the [DEFINE LISTENER](#).

Related concepts

[“Using the TCP listener backlog option on z/OS” on page 1052](#)

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.

Using the IBM MQ listener

To run the listener supplied with IBM MQ, which starts new channels as threads, use the `runmqclsr` command.

For example:

```
runmqclsr -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 263](#). 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:

```
endmqclsr [-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 AIX 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 249](#)) you must add the following entry to your queue manager configuration file (`qm.ini`):

```
TCP:
KeepAlive=yes
```

See the documentation for your AIX or Linux system for more information.

Linux AIX Defining an LU 6.2 connection on AIX 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 25. Settings on the local AIX or 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.	-
AIX and Linux systems	The same as the corresponding TPName in the side information about the remote queue manager.	<code>MQ_INSTALLATION_PATH/bin/amqcrs6a</code>
Windows	As specified in the Windows Run Listener command, or the invocable Transaction Program that was defined using TpSetup on Windows.	<code>MQ_INSTALLATION_PATH\bin\amqcrs6a</code>

`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 LU 6.2 on AIX and Linux” on page 286](#)

On AIX 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 AIX and Linux” on page 287](#)

On AIX and Linux systems, create a listening attachment at the receiving end, an LU 6.2 logical connection profile, and a TPN profile.

Linux AIX Sending end on LU 6.2 on AIX and Linux

On AIX 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
```

SECURITY PROGRAM is used, where supported by CPI-C, when IBM MQ attempts to establish an SNA session.

Linux AIX Receiving on LU 6.2 on AIX and Linux

On AIX 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	<code>MQ_INSTALLATION_PATH/bin/amqcrs6a</code>
Transaction Program name	<code>recv</code>
User ID	<code>0</code>

`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.

AIX On AIX, 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.

IBM i 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.

About this task

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 212](#).

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.

Procedure

- For more information about creating and working with objects, see:
 - [“Creating objects on IBM i” on page 288](#)
 - [“Creating a channel on IBM i” on page 288](#)
 - [“Starting a channel on IBM i” on page 291](#)
 - [“Selecting a channel on IBM i” on page 291](#)
 - [“Browsing a channel on IBM i” on page 292](#)
 - [“Renaming a channel on IBM i” on page 294](#)
 - [“Work with channel status on IBM i” on page 294](#)
 - [“Work-with-channel choices on IBM i” on page 295](#)

Related concepts

[“Setting up communication for IBM i” on page 301](#)

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.

Related tasks

[“Configuring connections between the client and server” on page 16](#)

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](#)

Creating objects on IBM i

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 [System and default objects](#).

Creating a channel on IBM i

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 . . . . . 99999999__ 100-99999999, *SYSDFTCHL
Maximum message length . . . . . 4194304___ 0-4194304, *SYSDFTCHL
Heartbeat interval . . . . . 300_____ 0-99999999, *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)

▶ IBM i Starting a channel on IBM i

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.

▶ IBM i Selecting a channel on IBM i

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 on IBM i

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.2T01
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 on IBM i

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 on IBM i

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.

IBM i Work-with-channel choices on IBM i

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 296	Changes the attributes of a channel.
"3=Copy" on page 296	Copies the attributes of a channel to a new channel.
"4=Delete" on page 296	Deletes a channel.
"5=Display" on page 296	Displays the current settings for the channel.
"6=Create" on page 296	Displays the Create channel panel
"8=Work with Status" on page 297	Displays the channel status panels.
"13=Ping" on page 298	Runs the Ping facility to test the connection to the adjacent system by exchanging a fixed data message with the remote end.

Menu option	Description
“14=Start” on page 298	Starts the selected channel, or resets a disabled receiver channel.
“15=End” on page 299	Requests the channel to close down.
“16=Reset” on page 300	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 300	Requests the channel to resolve in-doubt messages without establishing connection to the other end.
“18=Display authority” on page 300	Displays IBM MQ object authority
“19=Grant authority” on page 300	Grants IBM MQ object authority
“20=Revoke authority” on page 300	Revokes IBM MQ object authority
“21=Recover object” on page 301	Recovers IBM MQ object
“22=Record image” on page 301	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 i object names; see [Administering IBM MQ for IBM i](#).

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 289](#).

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 IBM i” on page 294](#).

- 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).

IBM i **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 IBM MQ Explorer.
-  On z/OS, by using the Start a channel panel.
-  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.

 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.

 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.

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 GRMQMAUT 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 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 either of the following two forms of communication between IBM MQ for IBM i systems:

- [“Defining a TCP connection on IBM i” on page 302](#)

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 302](#)

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.

Where needed, the triggering arrangement must also be prepared with the definition of the necessary processes and queues.

  A message channel that uses TCP/IP can be pointed at an IBM Aspera faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. A queue manager running on any entitled platform can connect through an Aspera gateway. The gateway itself is deployed on Red Hat or Ubuntu Linux, or Windows. See [Defining an Aspera gateway connection on Linux or Windows](#).

Related tasks

[“Monitoring and controlling channels on IBM i” on page 287](#)

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.

  A message channel that uses TCP/IP can be pointed at an IBM Aspera faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. A queue manager running on any entitled platform can connect through an Aspera gateway. The gateway itself is deployed on Red Hat or Ubuntu Linux, or Windows. See [Defining an Aspera gateway connection on Linux or Windows](#).

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 IBM MQ for Multiplatforms” on page 284](#) for more information, and the specific value for IBM i.

Related concepts

[“Receiving on TCP” on page 302](#)

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 249](#)) 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 26. 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.
AIX 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 304](#)

Use the CRTMQMCHL command to define a channel of transport type *LU62.

[“Initiated end \(Receiver\)” on page 307](#)

Use the CRTMQMCHL command to define the receiving end of the message channel link with transport type *LU62.

IBM i 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.

IBM i *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 'QZRCRVR' 37
30 *RTGDTA 'QZHQTRG' 37
50 *RTGDTA 'QVPPRINT' 37
60 *RTGDTA 'QNPSRVR' 37
70 *RTGDTA 'QNMAPINGD' 37
80 QNMAREXCD QSYS 'AREXCD' 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 452](#)

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 322](#)

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 333](#)

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.

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 MQ00_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 Multiplatforms queue manager. 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 to 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](#)

[How to choose what type of cluster transmission queue to use](#)

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.

Multi On Multiplatforms, 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** MQSC 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 [Multiplatforms](#) 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
- If the cluster needs to use **PROPCTL** to remove application headers such as RFH2 from messages going from an IBM MQ queue manager to a queue manager on an earlier version of the product, 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.

 **Multi** On Multiplatforms, you can set a default local address value that is 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 , 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 315](#) describes the process of selecting a default transmission queue; [“Selection of manually defined transmission queues by cluster-sender channels” on page 315](#) 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 316](#) 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. 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 365.

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 315](#). 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`.

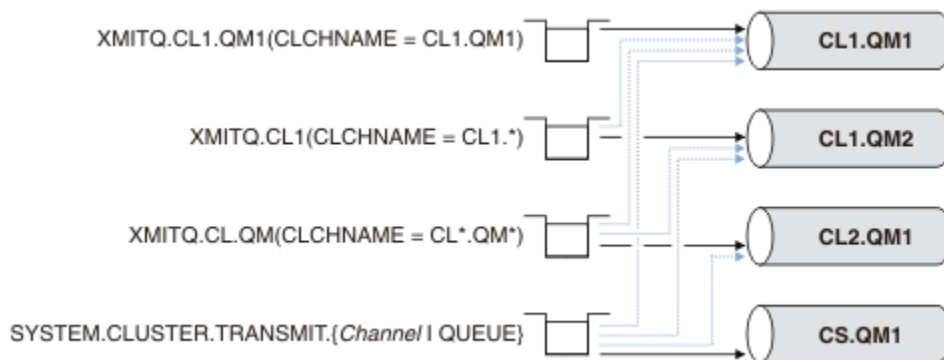


Figure 37. Transmission queue / cluster-sender channel precedence

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 315 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 315 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 319, 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 319, 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.

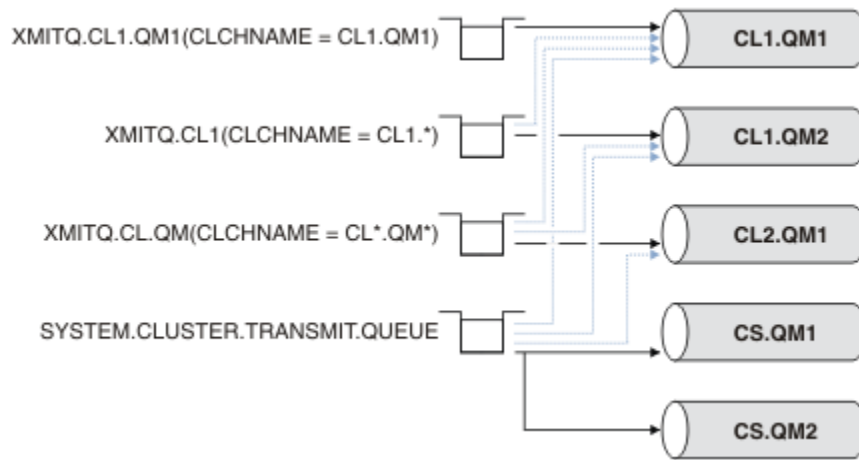


Figure 38. Multiple cluster sender channels

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 316.

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 733

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 355](#)

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 335](#)

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 362](#)

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 365](#)

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.

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
-  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 Multiplatforms, 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`.

 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** On 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.
- ▶ **ALW** On AIX, Linux, and Windows, 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** On 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** On 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 **CONNNAME** 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** On 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** On 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** ▶ **AIX** On AIX and Linux, 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 **runmqclsr** command. For example:

```
runmqclsr -t tcp -p 1414 -m QM1
```

To use **inetd** to start channels, configure two files:

- 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/amqcrista amqcrista  
-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:

AIX

On AIX:

```
refresh -s inetd
```

Linux

On Linux:

- a. Find the process ID of the **inetd** with the command:

```
ps -ef | grep inetd
```

- b. Run the following command:

```
kill -1 inetd processid
```

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.

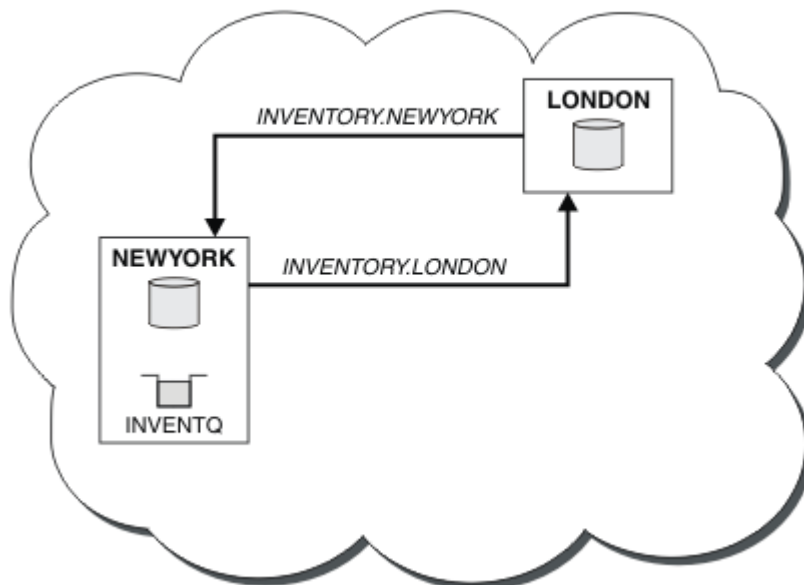
Before you begin

- Instead of following these instructions, you can use one of the wizards supplied with IBM 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 309, [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.

INVENTORY



This is what this cluster looks like:

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.

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.

Procedure

- [“Setting up a cluster using TCP/IP with a single transmission queue per queue manager” on page 323](#)
- [“Setting up a cluster on TCP/IP using multiple transmission queues per queue manager” on page 326](#)
-  [“Setting up a cluster using LU 6.2 on z/OS” on page 329](#)
- [“Verifying the cluster” on page 331](#)

Related concepts

[Clusters](#)

[Comparison of clustering and distributed queuing](#)

[Components of a cluster](#)

Related tasks

[“Configuring a queue manager cluster” on page 309](#)

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

This is one of three topics describing different configurations for a simple cluster.

Before you begin

For an overview of the cluster that is being created, see [“Setting up a new cluster” on page 322](#).

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 [Multiplatforms](#) 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 1050 to set up the TCP/IP connection, rather than defining the listeners in step [“4”](#) on page 324. 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 333.

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: IBM 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.

What to do next

You are now ready to [verify the cluster](#).

Related tasks

[“Setting up a cluster on TCP/IP using multiple transmission queues per queue manager” on page 326](#)

This is one of three topics describing different configurations for a simple cluster.

[“Setting up a cluster using LU 6.2 on z/OS” on page 329](#)

This is one of three topics describing different configurations for a simple cluster.

Setting up a cluster on TCP/IP using multiple transmission queues per queue manager

This is one of three topics describing different configurations for a simple cluster.

Before you begin

For an overview of the cluster that is being created, see [“Setting up a new cluster” on page 322](#).

About this task

Follow these steps to set up a cluster on [Multiplatforms](#) 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 335](#).

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 333](#).

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: IBM 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 333](#) and [“Adding a queue manager to a cluster: separate transmission queues” on page 335](#).

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.

What to do next

You are now ready to [verify the cluster](#).

Related tasks

[“Setting up a cluster using TCP/IP with a single transmission queue per queue manager” on page 323](#)

This is one of three topics describing different configurations for a simple cluster.

[“Setting up a cluster using LU 6.2 on z/OS” on page 329](#)

This is one of three topics describing different configurations for a simple cluster.

Setting up a cluster using LU 6.2 on z/OS

This is one of three topics describing different configurations for a simple cluster.

Before you begin

For an overview of the cluster that is being created, see [“Setting up a new cluster” on page 322](#).

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 333](#).

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 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: IBM 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.

 See [The channel initiator on z/OS](#) and [“Receiving on LU 6.2” on page 1054](#).

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.

What to do next

You are now ready to [verify the cluster](#).

Related tasks

[“Setting up a cluster using TCP/IP with a single transmission queue per queue manager” on page 323](#)
This is one of three topics describing different configurations for a simple cluster.

[“Setting up a cluster on TCP/IP using multiple transmission queues per queue manager” on page 326](#)
This is one of three topics describing different configurations for a simple cluster.

Verifying the cluster

Peer topics describe three different configurations for a simple cluster. This topic explains how to verify the cluster.

Before you begin

This topic assumes that you are verifying a cluster you created through one of the following tasks:

- [“Setting up a cluster using TCP/IP with a single transmission queue per queue manager” on page 323](#).
- [“Setting up a cluster on TCP/IP using multiple transmission queues per queue manager” on page 326](#).
-  [“Setting up a cluster using LU 6.2 on z/OS” on page 329](#).

For an overview of the cluster that has been created, see [“Setting up a new cluster” on page 322](#).

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 **amqspout**.

3. On LONDON run the command **amqspout 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 322. 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.  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.

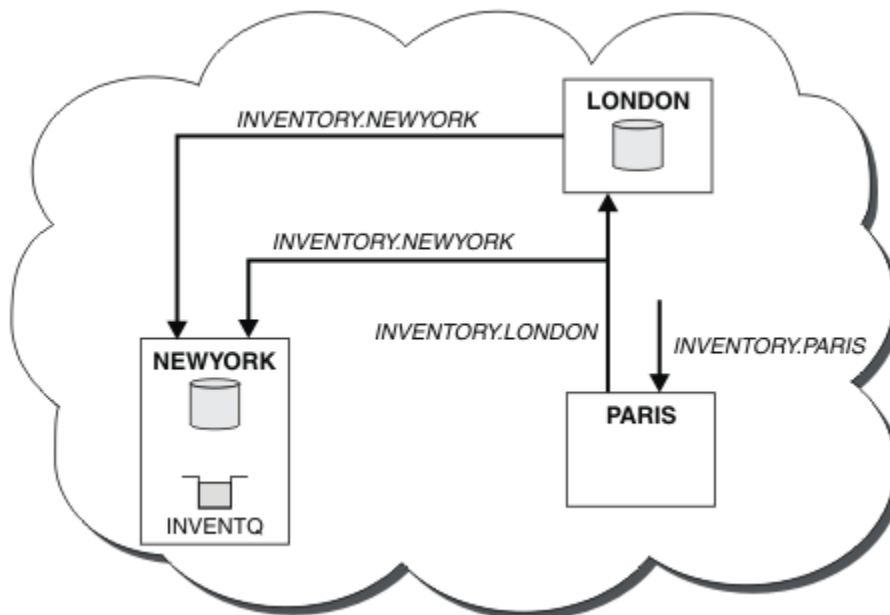


Figure 39. The INVENTORY cluster with three queue managers

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 322](#).

About this task

This task is an alternative to [“Adding a queue manager to a cluster” on page 333](#), 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.

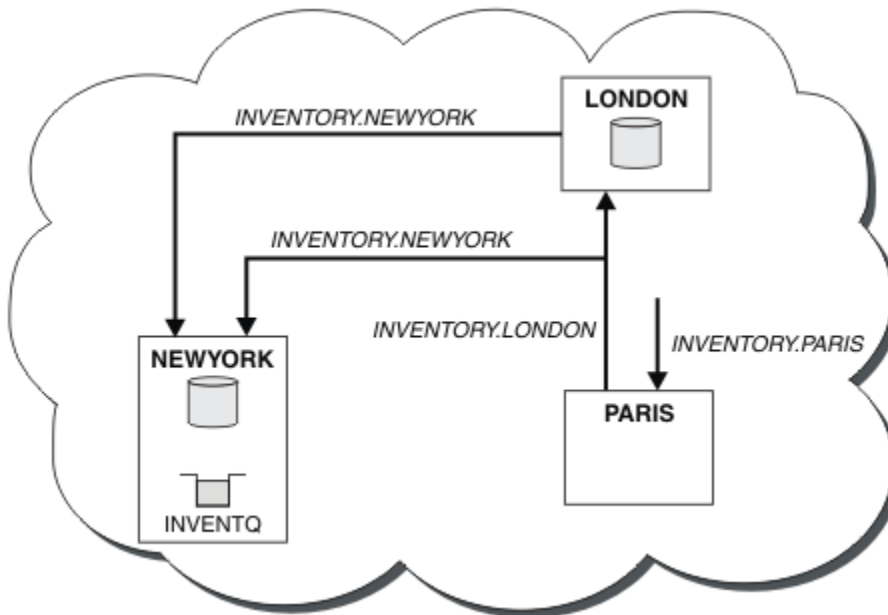


Figure 40. The INVENTORY cluster with three queue managers

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.

Related concepts

[How to choose what type of cluster transmission queue to use](#)

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 322](#). 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 333:

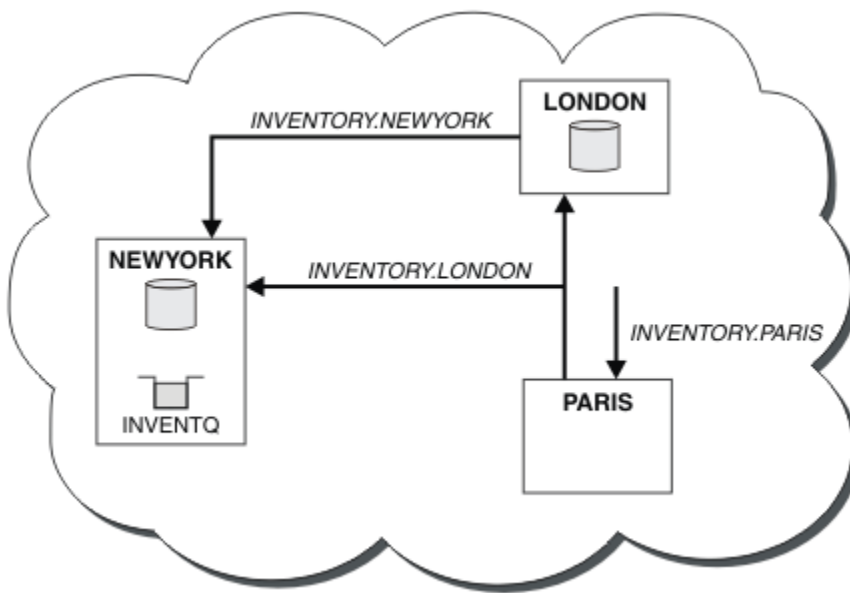


Figure 41. The INVENTORY cluster with three queue managers

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 333. 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 344.

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 42 on page 341 shows the INVENTORY cluster set up by this task.

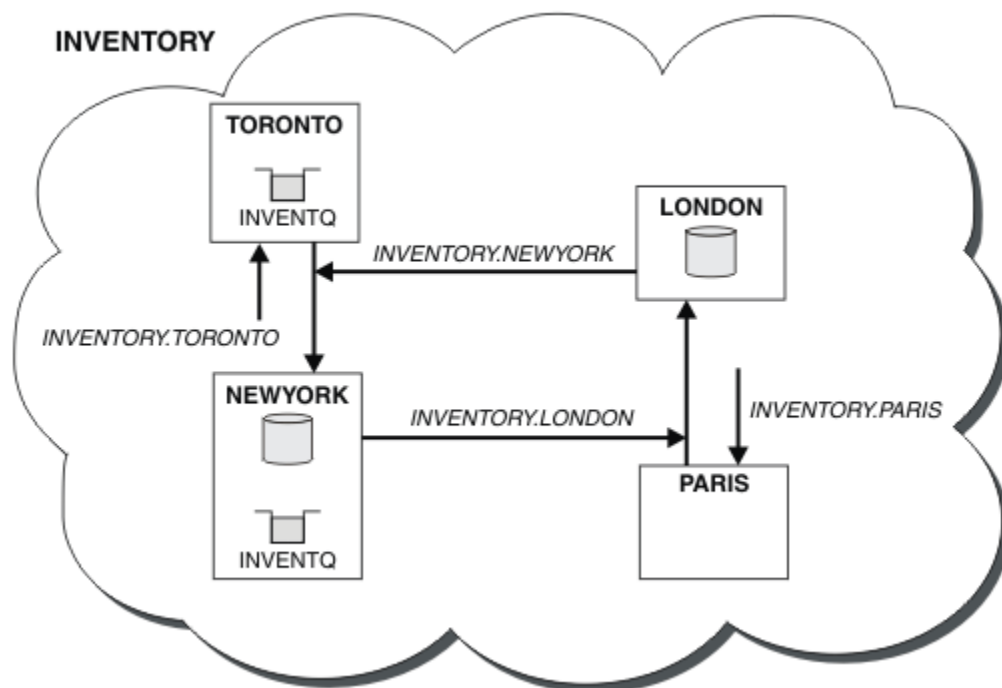


Figure 42. The INVENTORY cluster with four queue managers

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 322](#). 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 43 on page 343 shows the cluster set up by this task.

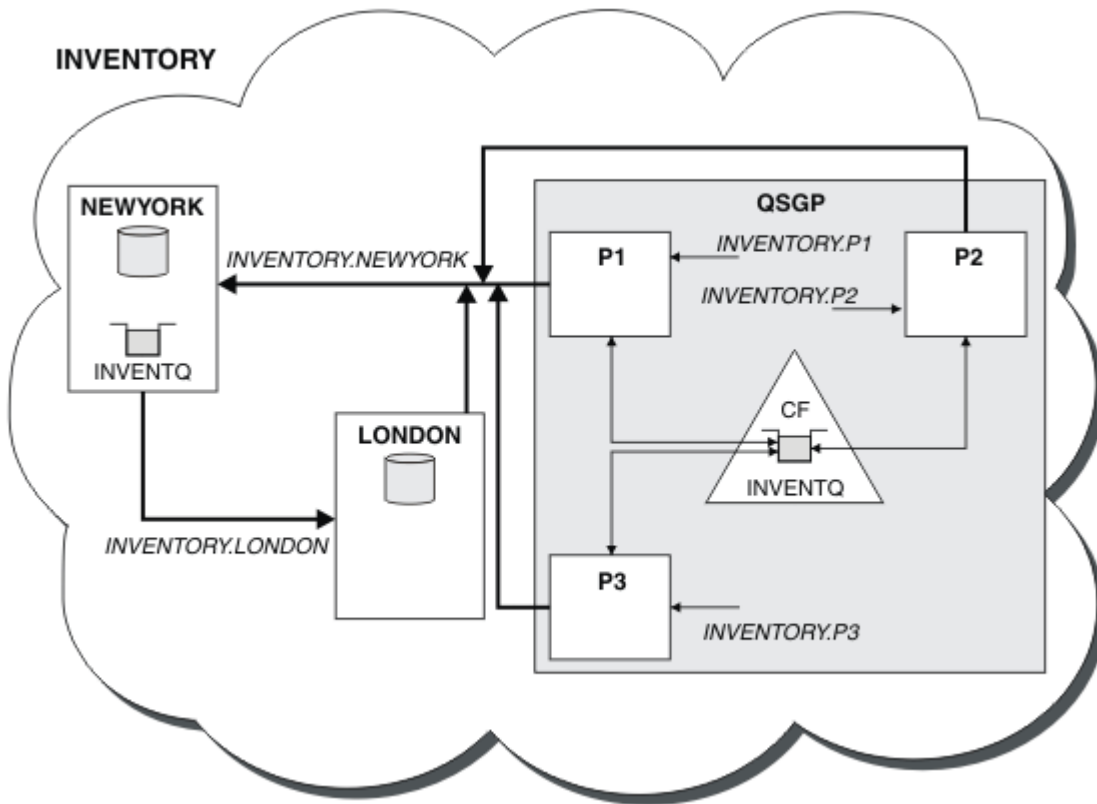


Figure 43. Cluster and queue sharing group

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 333.
- 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 44 on page 346 shows the cluster set up by this task.

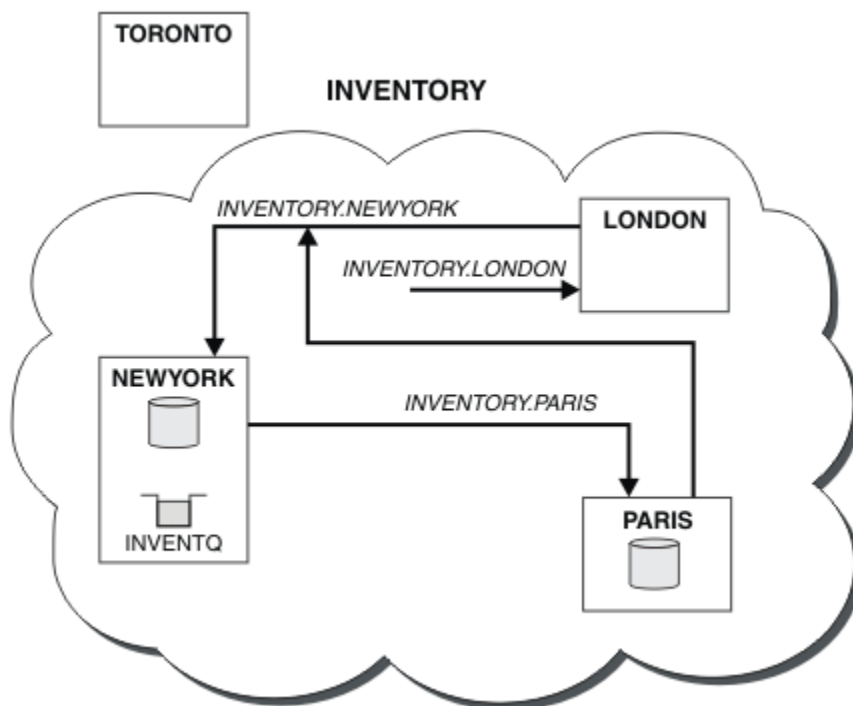


Figure 44. The INVENTORY cluster with the full repository moved to PARIS

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 322 through “Moving a full repository to another queue manager” on page 344 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 45 on page 347.

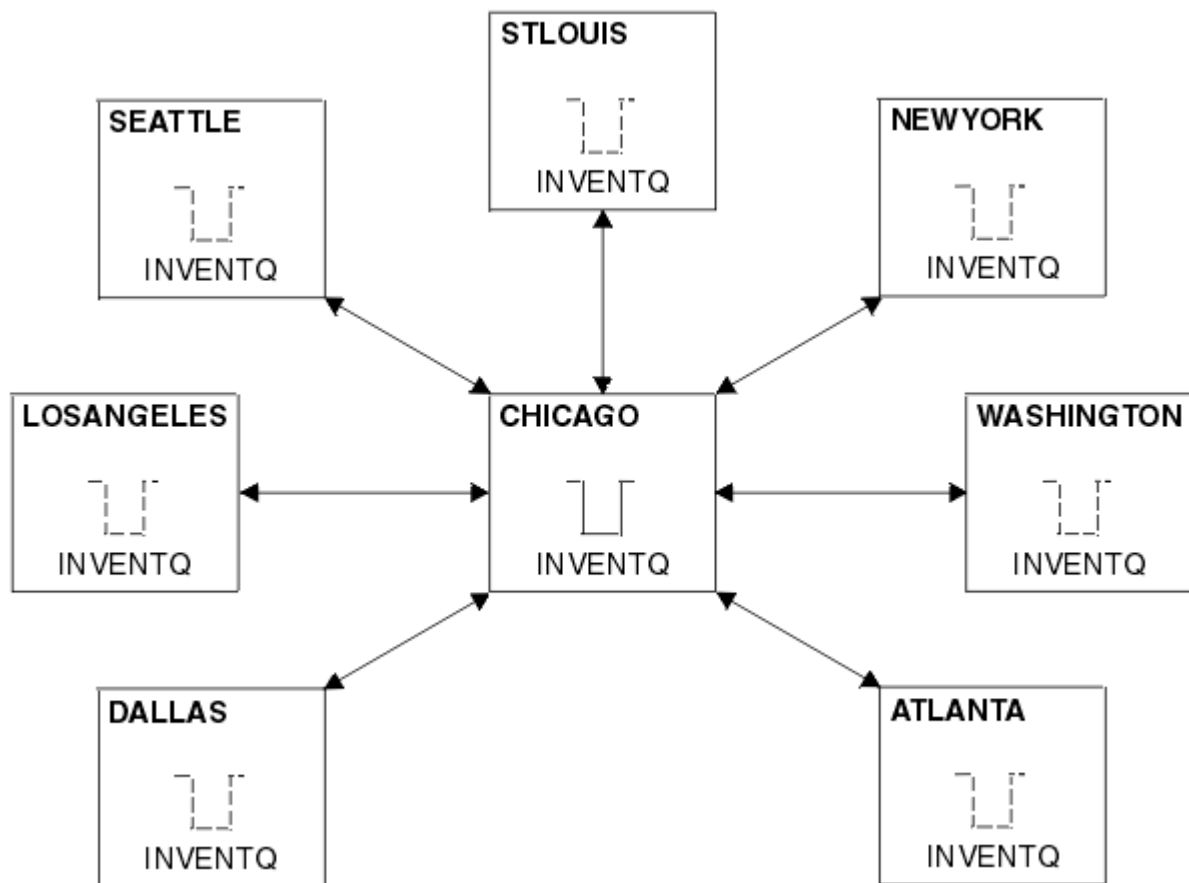


Figure 45. A hub and spoke network

- 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_name* 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 421](#)

2. Alter the two central queue managers to make them full repository queue managers.

The two queue managers CHICAGO and CHICAG02 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 CHICAG02:

```
DEFINE CHANNEL(CHNSTORE.CHICAG02) CHLTYPE(CLUSSDR) TRPTYPE(TCP)
CONNAME(CHICAG02.CHSTORE.COM) CLUSTER(CHNSTORE)
DESCR('Cluster-sender channel to repository queue manager')
```

5. Install the inventory application on CHICAG02.

You already have the inventory application on queue manager CHICAGO. Now you need to make a copy of this application on queue manager CHICAG02.

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 CHICAG02, 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 CHICAG02 of the INVENTQ on CHICAGO.

When you make these definitions, a message is sent to the full repositories at CHICAGO and CHICAG02 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 through 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 346.
- 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 through 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 46 on page 352](#).

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.

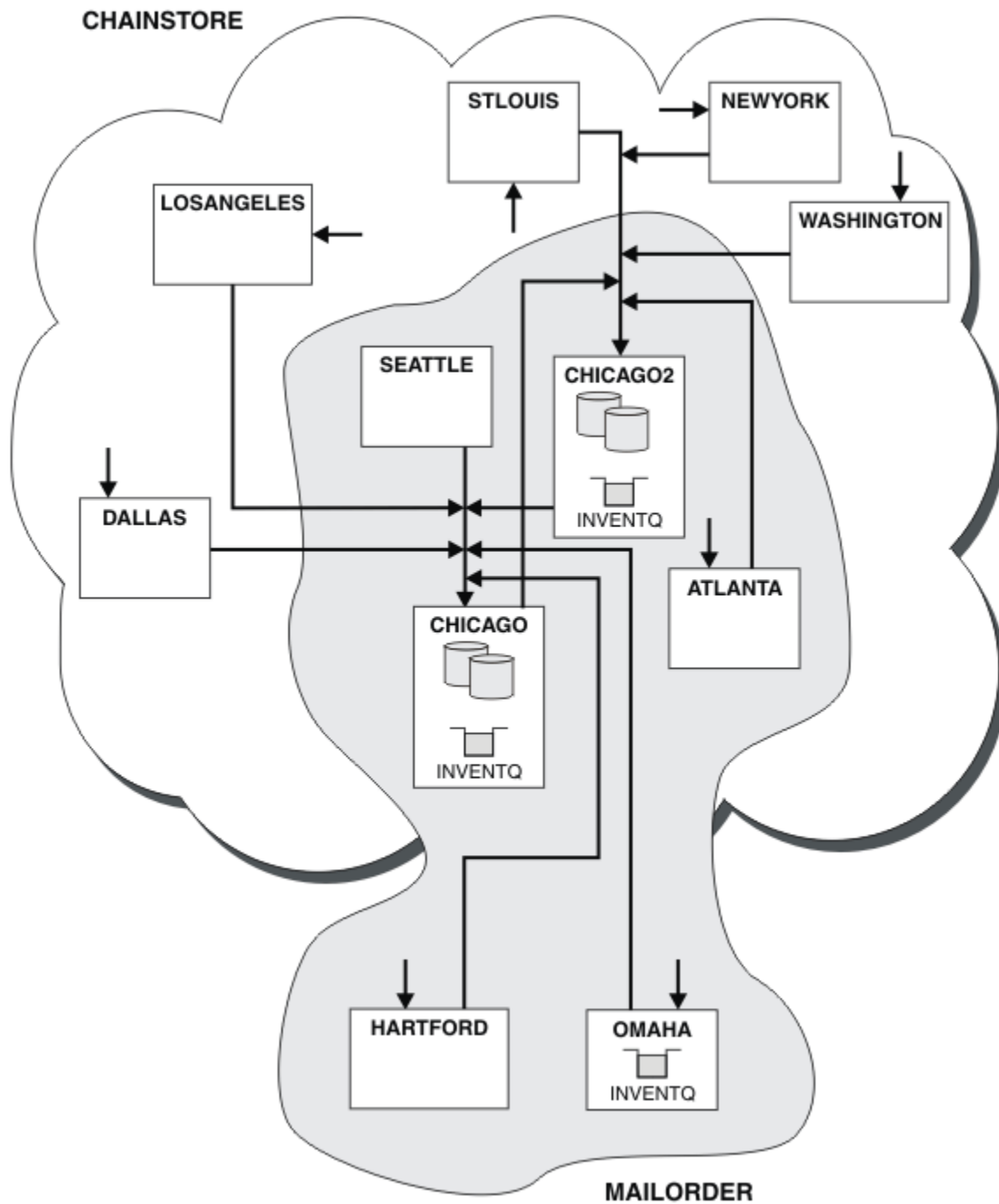


Figure 46. Interconnected clusters

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 346.
- 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 381](#).

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 47 on page 355](#). The example is described in [Clustering: Application isolation using multiple cluster transmission queues](#).

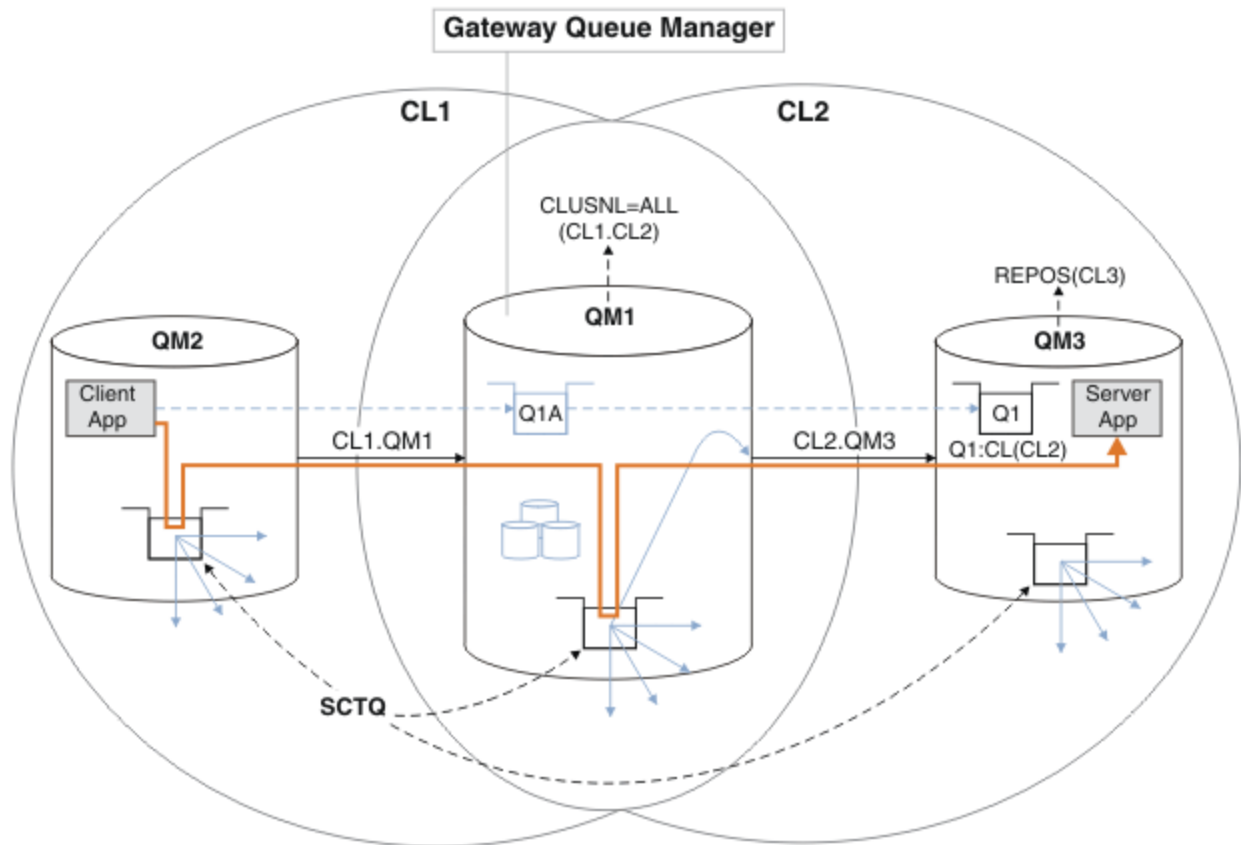


Figure 47. Client-server application deployed to hub and spoke architecture using IBM MQ clusters

To minimize the number of steps to construct the example, 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 additional queue managers so that each cluster has two repositories. The gateway queue manager **QM1** is not used as a repository for performance reasons.

Procedure

1. Create and start the gateway queue manager **QM1**, and queue managers **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 queue manager.

```
*... 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 gateway queue manager 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 27 on page 356 for each queue manager:

Table 27. Parameter values for creating clusters 1 and 2			
Queue manager	Cluster <i>c</i>	Other repository <i>n</i>	This repository <i>m</i>
QM2	1	4	2
QM4	1	2	4
QM3	2	5	3
QM5	2	3	5

```
*... On QMm
DEFINE CHANNEL(Clc.QMn) CHLTYPE(CLUSSDR) CONNAME('localhost(141n)') CLUSTER(Clc) REPLACE
DEFINE CHANNEL(Clc.QMm) CHLTYPE(CLUSRCVR) CONNAME('localhost(141m)') CLUSTER(Clc) REPLACE
```

6. Add the gateway queue manager QM1 to each of the clusters.

```
*... On gateway queue manager 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 a queue alias on the gateway queue manager QM1 for each clustered queue.

```
*... On gateway queue manager QM1
DEFINE QALIAS(Q1A) CLUSNL(ALL) TARGET(Q1) TARGTYPE(Queue) DEFBIND(NOTFIXED) REPLACE
```


9. Add the cluster queue manager alias definitions, for all the clustered queue managers, to the gateway queue manager QM1.

A queue manager alias is needed for each Partial Repository to which a response (Reply) will be needed. The purpose of the queue manager alias is this: If an application is processing a request message that includes a **ReplyTo** attribute (that originated from a queue manager in a different cluster), the application can reply to the request through the queue manager alias.

Note: In the following example, the queue manager alias names are QM2 and QM3, which are the same as the real queue manager names (**RQMNAME**) of QM2 and QM3 they refer to. If you want the applications not to know the real queue manager names, you can use different names. For example, the queue manager alias could be named PAYROLLQM, if this is more meaningful to the application than QM2.

```
*... On gateway queue manager 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 QM1 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 **amqspout** on QM2 to put a message.

```
C:\IBM\MQ>amqspout 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 gateway queue manager QM1. QM3 routes the reply message to gateway queue manager QM1. QM1 routes the message to QM2.

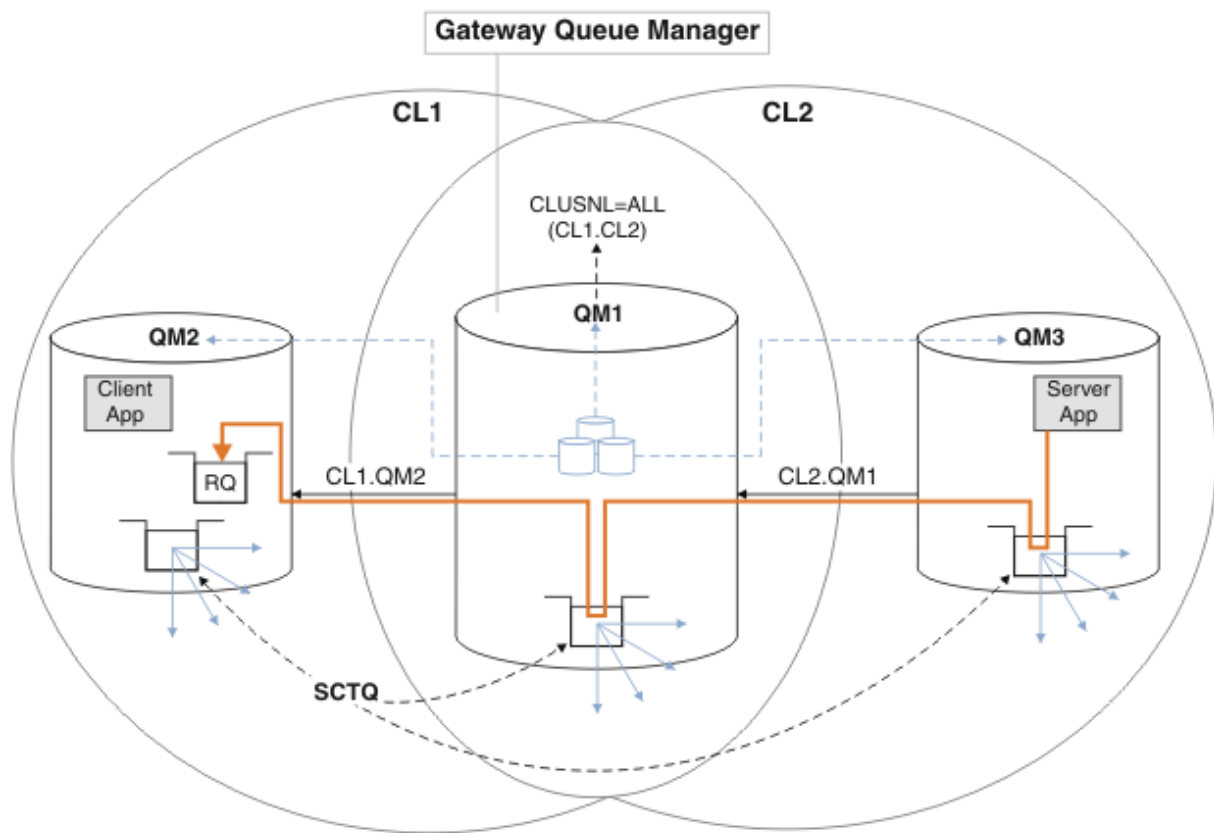


Figure 48. Using a queue manager alias to return the reply message to a different cluster

The way the routing works is as follows. Every queue manager in each cluster has a queue manager alias definition on gateway queue manager 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 gateway queue manager 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.

Gateway queue manager QM1 places the message on its cluster transmission queue for transfer to QM2. Gateway queue manager 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 gateway queue manager QM1, because both QM1 and QM2 are in the same cluster, CL1. Gateway queue manager 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 gateway queue manager 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, 2026. 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 335](#)

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 355 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 49 on page 360, 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 362. You also have to manage the switch from one transmission queue to the other.

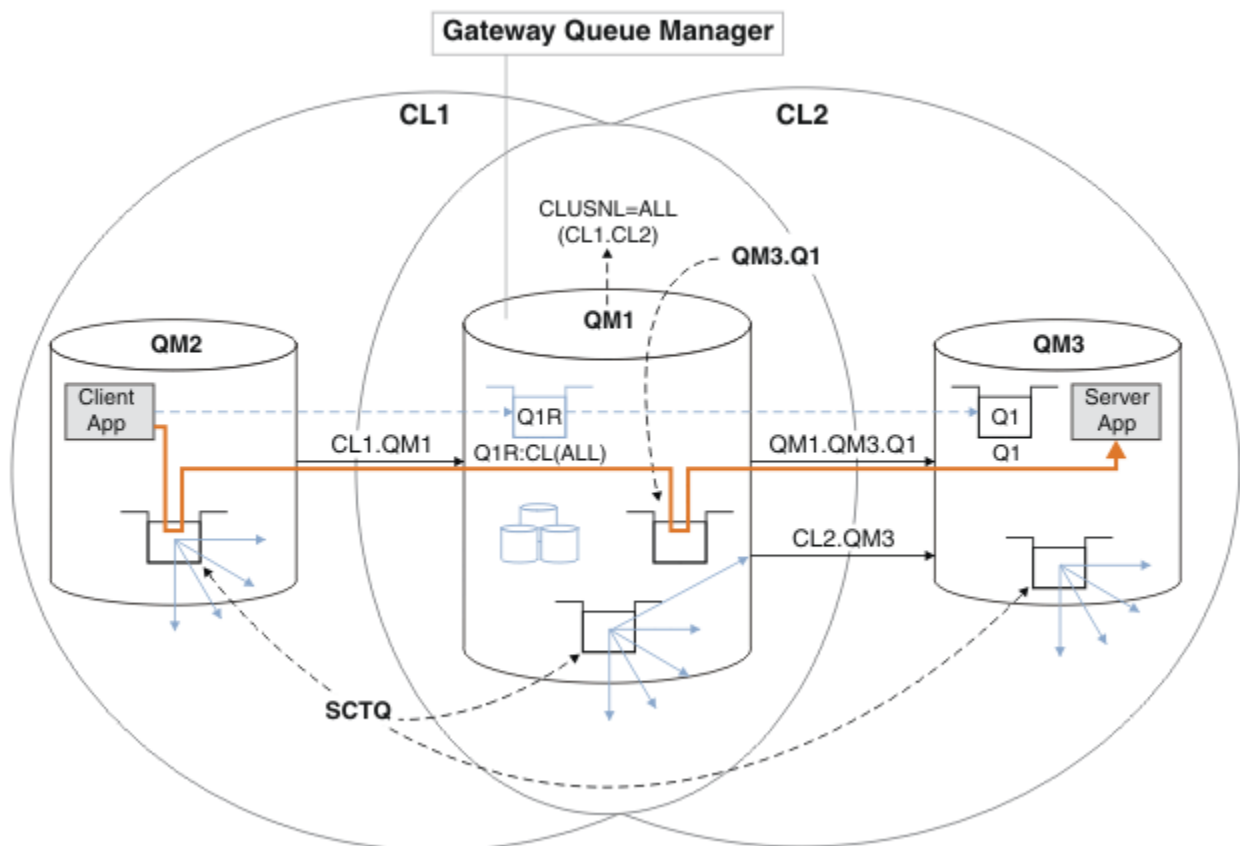


Figure 49. Client-server application deployed to hub and spoke cluster architecture using remote queue definitions

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 **amqspu**t on QM2 to put a message.

```
C:\IBM\MQ>amqspu 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 **amqsge**t to get the message from Q1 on QM3

```
C:\IBM\MQ>amqsge 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 335

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 IBM MQ.
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 355 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 50 on page 363.

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 359:

- 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 losing 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 50 on page 363 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 359. 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 50 on page 363 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 365.

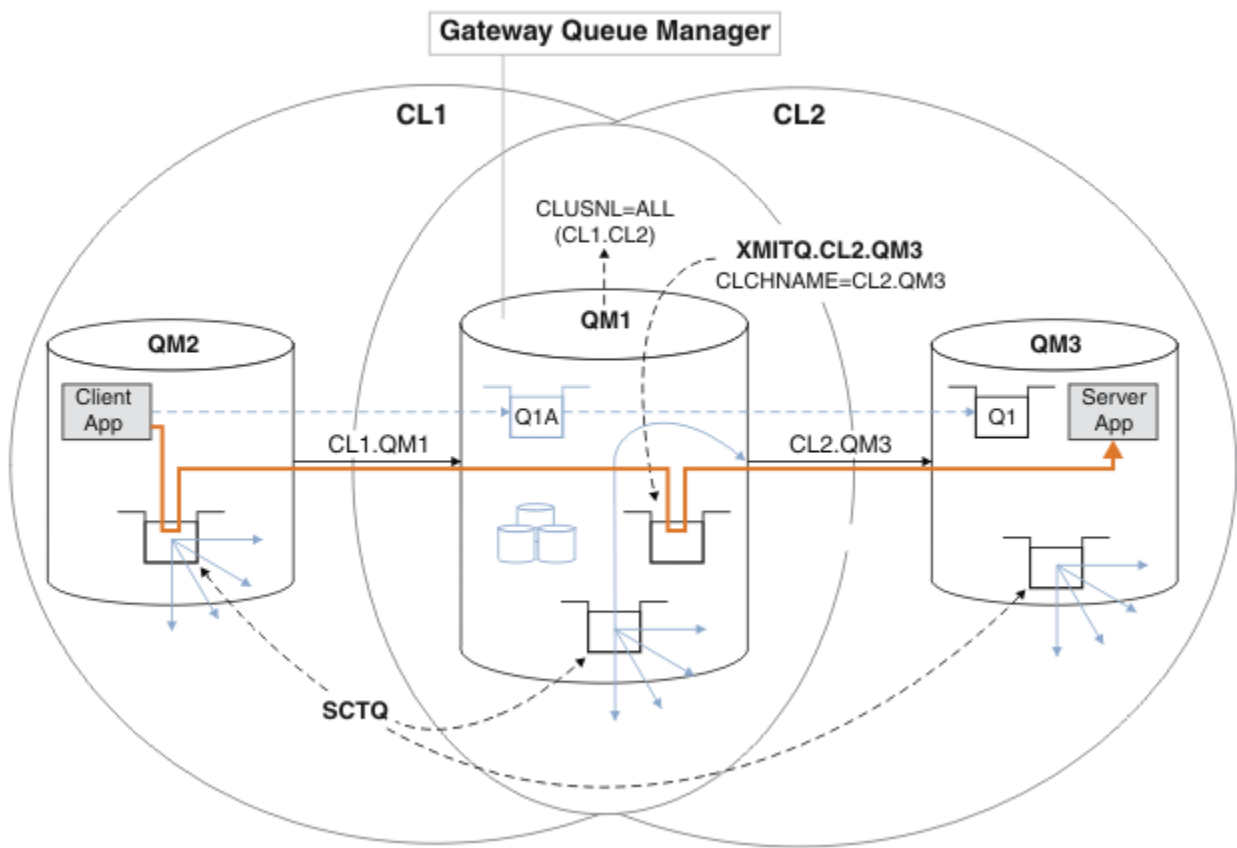


Figure 50. Client-server application deployed to hub and spoke architecture using an additional cluster transmission queue.

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)
CONNNAME(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)
CONNNAME(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 **amqspout** on QM2 to put a message.

```
C:\IBM\MQ>amqspout 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 314](#)

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 335](#)

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 50 on page 363](#).

1. The gateway queue manager must be on IBM MQ.
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 355](#) by following the steps in that task.
3. Do the steps in [Figure 50 on page 363](#) in [“Adding a cluster transmit queue to isolate cluster message traffic sent from a gateway queue manager” on page 362](#) 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 362](#) 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 362 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 51 on page 366. 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 362 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.

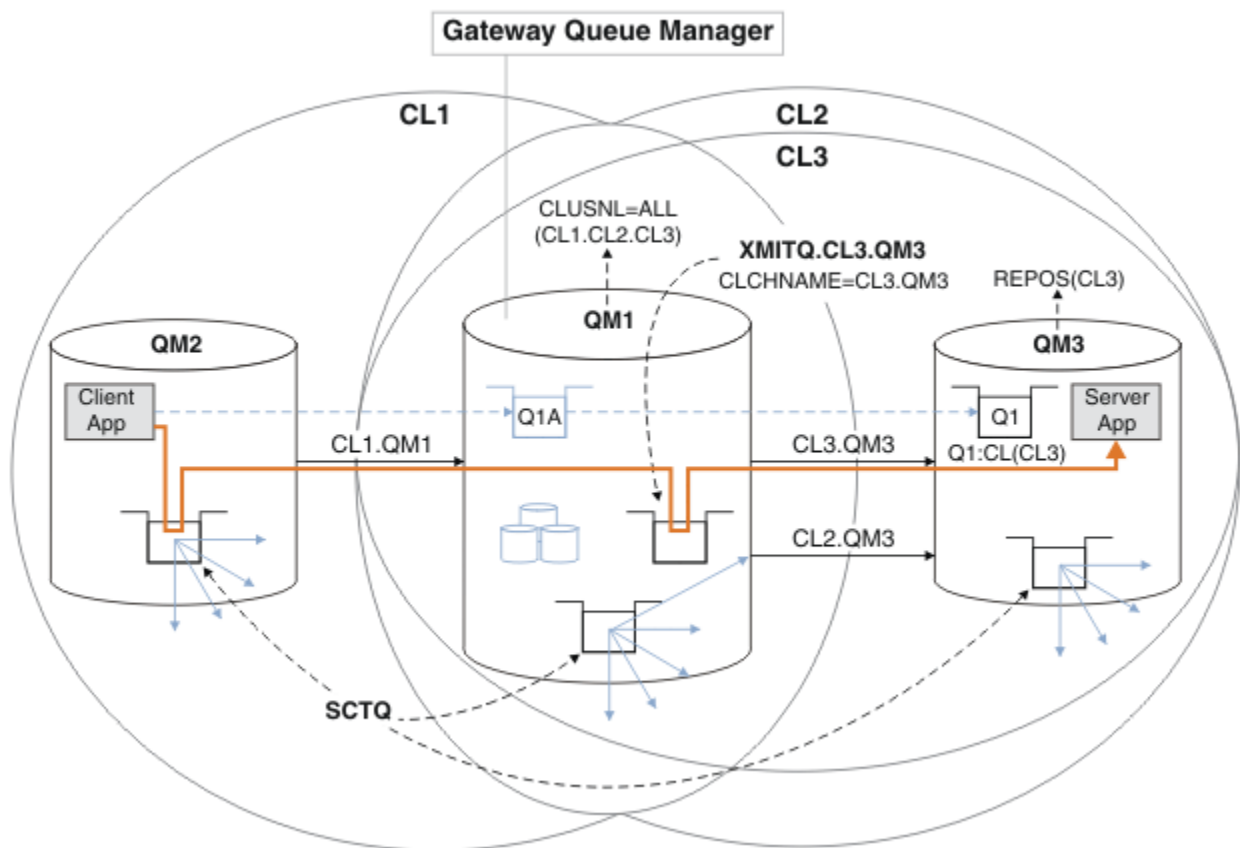


Figure 51. Using an additional cluster to separate message traffic in the gateway queue manager that goes to one of a number of cluster queues on the same queue manager

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 **amqspout** on QM2 to put a message.

```
C:\IBM\MQ>amqspout 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 314](#)

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 335

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 IBM MQ.
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 355 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 IBM MQ. 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 52 on page 371. 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 335.

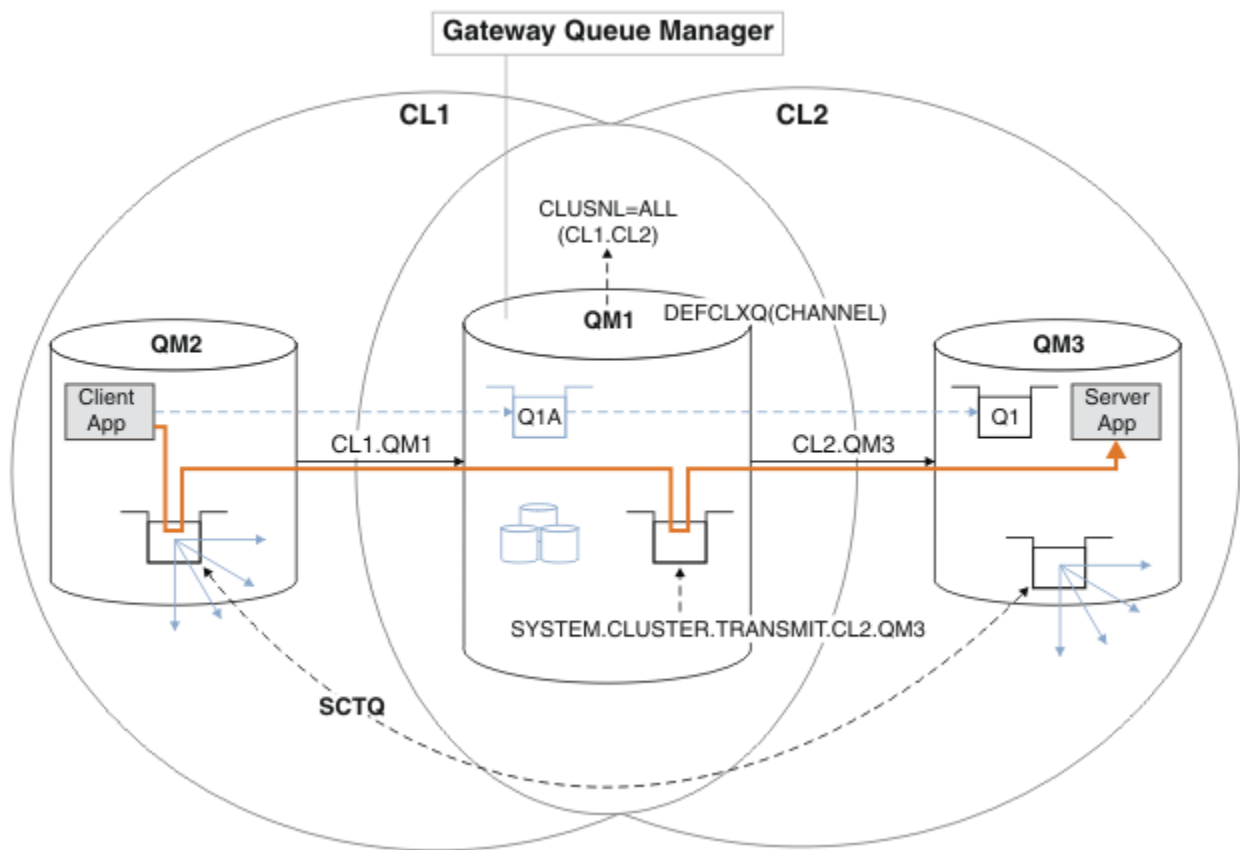


Figure 52. Client-server application deployed to hub and spoke architecture with separate cluster transmission queues on the gateway queue manager.

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(QLLOCAL)
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 (CL1.0M2) CHLTYPE (CLUSSDR)

```
CONNAME(127.0.0.1(1412))
```

ROMNAME(OM2) STATUS(RUNNING)

SUBSTATE (MOGET)

```
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL1.QM2)
```

AMQ8417: Display Channel Status details.

CHANNEL (CL2.QM3) CHLTYPE (CLUSSDR)

```
CONNAME(127.0.0.1(1413))
```

RQMNAME(QM3) STATUS(RUNNING)

SUBSTATE (MOGET)

```
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 (MOGET)

```
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL2.QM5)
```

AMQ8417: Display Channel Status details.

CHANNEL (CL1.QM4) CHLTYPE (CLUSSDR)

```
CONNAME(127.0.0.1(1414))
```

```

ROMNAME(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 **amqspout** on QM2 to put a message.

```
C:\IBM\MQ>amqspout 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 335](#)

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 340](#). 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 queues are 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 set up 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: best practice

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 379.

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 333, and modified as described in [“Removing a cluster queue from a queue manager”](#) on page 375.

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 344. 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 375.
 - 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 452.

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.PARIS) 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)
```

If a manually defined transmission queue was in use for this channel, and the CLCHNAME pattern does not match any other existing or planned channels, you might want to delete the transmission queue. For example:

```
DELETE QLOCAL(PARIS.CUSTOM.XMITQ)
```

Note: If auto-defined transmission queues or the shared SYSTEM.CLUSTER.TRANSMIT.QUEUE are in use, this step is not required.

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 333](#). 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 381](#).

Related tasks

[Removing a queue manager from a cluster \(using IBM MQ Explorer\)](#)

Related reference

[ALTER CHANNEL \(alter channel settings\)](#)

[DISPLAY CHANNEL \(display channel definition\)](#)

[DISPLAY CHSTATUS \(display channel status\)](#)

[DISPLAY CLUSQMGR \(display channel information for cluster queue managers\)](#)

[STOP CHANNEL \(stop a channel\)](#)

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: best practice” on page 377](#).

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 333](#), and modified as described in [“Removing a cluster queue from a queue manager” on page 375](#).

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 344. 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 375.
- 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 452.

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 333. 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 381.

Related reference

[DELETE CHANNEL \(delete a channel\)](#)

[DISPLAY CHANNEL \(display channel definition\)](#)

[DISPLAY CHSTATUS \(display channel status\)](#)

[DISPLAY CLUSQMGR \(display channel information for cluster queue managers\)](#)

[STOP CHANNEL \(stop a channel\)](#)

[RESET CLUSTER \(reset a cluster\)](#)

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: best practice”](#) on page 377. 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](#)

[Cluster components](#)

Related reference

[Comparison of clustering and distributed queuing](#)

Maintaining a queue manager in a cluster

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(clusterName)
```

Where *clusterName* specifies the name of the cluster.

The `SUSPEND runmqsc` command notifies the queue managers in the 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(clusterName) MODE(FORCE)
```

Where *clusterName* specifies the name of the cluster.

`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.

For more information, see [SUSPEND QMGR](#).

2. Check that no messages are sent to the queue manager.

You must close any application that continues to send messages to this queue manager. The cluster workload algorithm might choose the suspended queue manager. If there are no other valid destinations, or if an application has an affinity with the queue manager, it might select the queue manager.

3. Do whatever maintenance tasks are necessary.

4. Resume the queue manager by issuing the RESUME QMGR **runmqsc** command:

```
RESUME QMGR CLUSTER(clusterName)
```

Where *clusterName* specifies the name of the cluster.

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.  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  (for example z/OS page sets) does not become full.

About this task



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 IBM Support. 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 concepts

[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 KAJINT 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.

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 Multiplatforms

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 Registry.

Example

See “[Suggested settings](#)” on page 251 for examples of how you implement these settings on all platforms.

Checking that async commands for distributed networks have finished

Many commands are asynchronous when used in a distributed network. Depending on the command, and the network state when it is issued, it can take a significant amount of time to finish. The queue manager does not issue a message on completion, so you need other ways of checking that the command has finished.

About this task

Almost any configuration change that you make to a cluster is likely to complete asynchronously. This is because of the internal administration and updating cycles that operate within clusters. For publish/subscribe hierarchies, any configuration change that affects subscriptions is likely to complete asynchronously. This is not always obvious from the name of the command.

The following MQSC commands might all complete asynchronously. Each of these commands has a PCF equivalent, and most are also available from within IBM MQ Explorer . When run on a small network with no workload, these commands typically complete within a few seconds. However, this is not the case for larger and busier networks. Also, the **REFRESH CLUSTER** command might take much longer, particularly when it is issued on multiple queue managers at the same time.

To have confidence that these commands have finished, check that the expected objects exist on the remote queue managers.

Procedure

- ALTER QMGR

For the ALTER QMGR PARENT command, use `DISPLAY PUBSUB TYPE(PARENT) ALL` to track the status of the requested parent relationship.

For the ALTER QMGR REPOS and ALTER QMGR REPOSNL commands, use `DISPLAY CLUSQMGR QMTYPE` to confirm completion.

- DEFINE CHANNEL, ALTER CHANNEL, and DELETE CHANNEL

For all parameters listed in the table ALTER CHANNEL parameters, use the `DISPLAY CLUSQMGR` command to monitor when changes have been propagated to the cluster.

- DEFINE NAMELIST, ALTER NAMELIST, and DELETE NAMELIST.

If you use a **NAMELIST** on the **CLUSNL** attribute of a **QMgr** object, a queue or a cluster channel might affect that object. Monitor as appropriate for the affected object.

Changes to `SYSTEM.QPUBSUB.QUEUE.NAMELIST` might affect creation or cancellation of proxy subscriptions in a publish/subscribe hierarchy. Use the `DISPLAY SUB SUBTYPE(PROXY)` command to monitor this.

- DEFINE queues, ALTER queues, and DELETE queues.

For all parameters listed in the table Parameters that can be returned by the DISPLAY QUEUE command, use the `DISPLAY QCLUSTER` command to monitor when changes have been propagated to the cluster.

- DEFINE SUB, and DELETE SUB

When you define the first subscription on a topic string, you might create proxy subscriptions in a publish/subscribe hierarchy or publish/subscribe cluster. Similarly, when you delete the last subscription on a topic string, you might cancel proxy subscriptions in a publish/subscribe hierarchy or publish/subscribe cluster.

To check that a command defining or deleting a subscription has finished, check whether or not the expected proxy subscription exists on other queue managers in the distributed network. If you are using *direct routing* in a cluster, check that the expected proxy subscription exists on the other partial repositories in the cluster. If you are using *topic host routing* in a cluster, check that the expected proxy subscription exists on the matching topic hosts. Use the following MQSC command:

```
DISPLAY SUB(*) SUBTYPE(PROXY)
```

Use the same check for the following equivalent subscribe and unsubscribe MQI calls, when they are issued in a cluster or hierarchy:

- Subscribe by using MQSUB.
- Unsubscribe by using MQCLOSE with `MQCO_REMOVE_SUB`.

- DEFINE TOPIC, ALTER TOPIC, and DELETE TOPIC

To check that a command defining, altering or deleting a clustered topic has finished, display the topic in the other partial repositories in the cluster (if you are using *direct routing*) or on the other topic hosts (if you are using *topic host routing*).

For all parameters listed in the table Parameters that can be returned by the DISPLAY TOPIC command, use the `DISPLAY TCLUSTER` command to monitor when changes have been propagated to the cluster.

Note:

- The **CLUSTER** parameter can affect creation or cancellation of proxy subscriptions in a publish/subscribe cluster.
- The **PROXYSUB** and **SUBSCOPE** parameters can affect creation or cancellation of proxy subscriptions in a publish/subscribe hierarchy or publish/subscribe cluster.

– Use the `DISPLAY SUB SUBTYPE(PROXYSUB)` command to monitor this.

- [REFRESH CLUSTER](#)

If you are running the **REFRESH CLUSTER** command, poll the cluster command queue depth. Wait for it to reach zero, and remain at zero, before looking for the objects.

1. Use the following MQSC command to check that the cluster command queue depth is zero.

```
DISPLAY QL(SYSTEM.CLUSTER.COMMAND.QUEUE) CURDEPTH
```

2. Repeat the check until the queue depth reaches zero, and remains at zero in the subsequent check.

The **REFRESH CLUSTER** command removes and re-creates objects, and in large configurations can take a significant time to complete. See [REFRESH CLUSTER considerations for publish/subscribe clusters](#).

- [REFRESH QMGR TYPE\(PROXYSUB\)](#)

To check that the **REFRESH QMGR TYPE(PROXYSUB)** command has finished, check that the proxy subscriptions have been corrected on other queue managers in the distributed network. If you are using *direct routing* in a cluster, check that the proxy subscriptions have been corrected on the other partial repositories in the cluster. If you are using *topic host routing* in a cluster, check that the expected proxy subscriptions have been corrected on the matching topic hosts. Use the following MQSC command:

```
DISPLAY SUB(*) SUBTYPE(PROXYSUB)
```

- [RESET CLUSTER](#)

To check that the **RESET CLUSTER** command has completed, use `DISPLAY CLUSQMGR`.

- [RESET QMGR TYPE\(PUBSUB\)](#)

To check that the **RESET QMGR** command has completed, use `DISPLAY PUBSUB TYPE(PARENT | CHILD)`.

Note: The **RESET QMGR** command might cause cancellation of proxy subscriptions in a publish/subscribe hierarchy or publish/subscribe cluster. Use the `DISPLAY SUB SUBTYPE(PROXYSUB)` command to monitor this.

- You might also want to monitor other system queues that, as and when commands complete, tend towards a queue depth of zero.

For example, you might want to monitor the `SYSTEM.INTER.QMGR.CONTROL` queue, and the `SYSTEM.INTER.QMGR.FANREQ` queue. See [Monitoring proxy subscription traffic in clusters](#) and [Balancing producers and consumers in publish/subscribe networks](#).

What to do next

If these checks do not confirm that an asynchronous command has finished, an error might have occurred. To investigate, first check the log for the queue manager on which the command was issued, then (for a cluster) check the cluster full repository logs.

Related reference

 [Asynchronous behavior of CLUSTER commands on z/OS](#)

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](#)

[Components of a cluster](#)

[“Queue manager aliases and clusters” on page 400](#)

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 403](#)

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 402](#)

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 309](#)

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 322](#)

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.

Related reference

[Comparison of clustering and distributed queuing](#)

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 53 on page 389](#) 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)
```

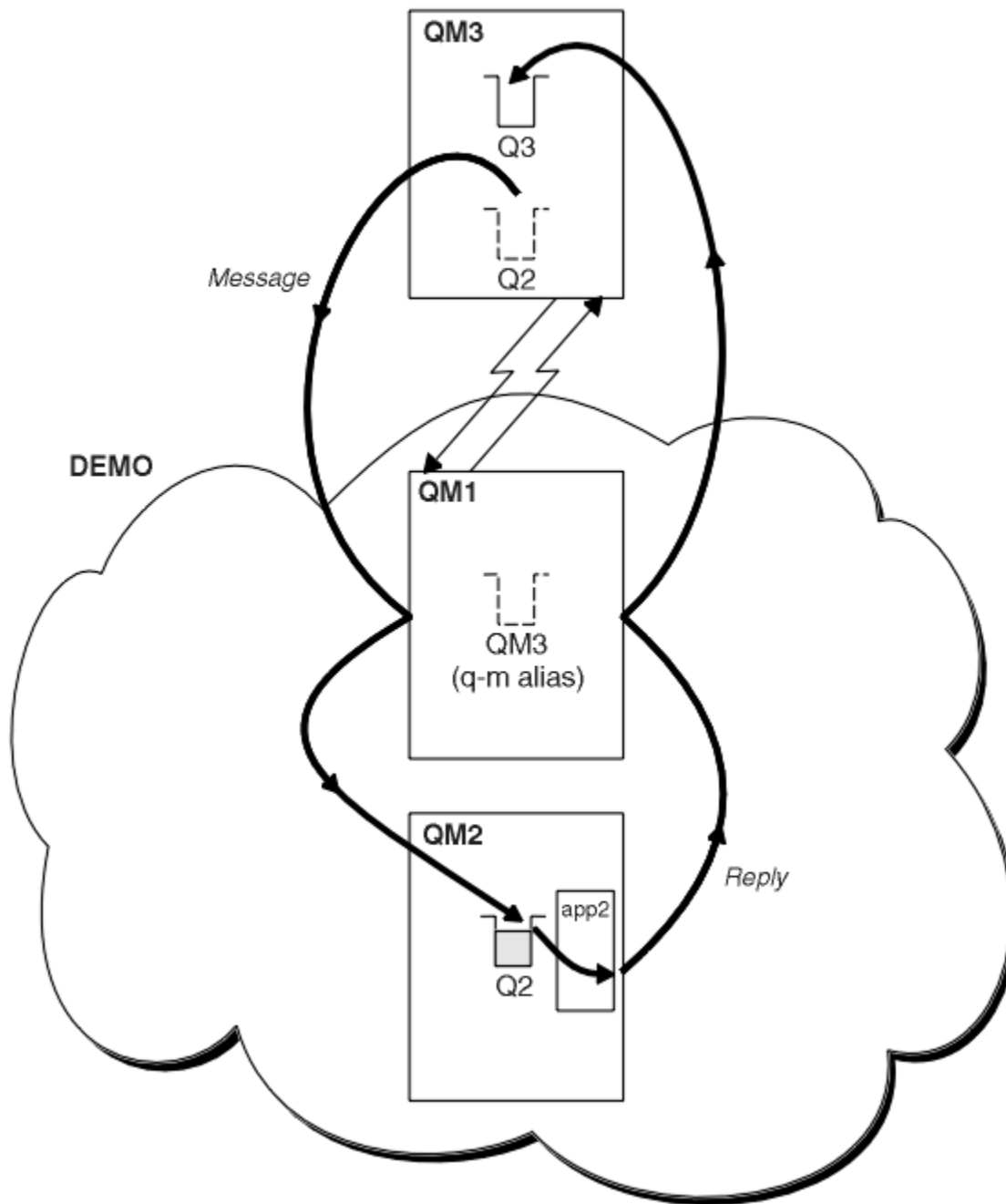



Figure 53. Putting from a queue manager outside the cluster

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 53 on page 389](#).

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 391](#)

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 391](#) 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 388](#), to hide the name of the target queue manager inside the cluster.

Procedure

In the example, see [Figure 54 on page 393](#), define a queue manager alias on the gateway queue manager QM1 called DEMO:

```
DEFINE QREMOTE(DEMO) RNAME(' ') RQMNAME(' ')
```

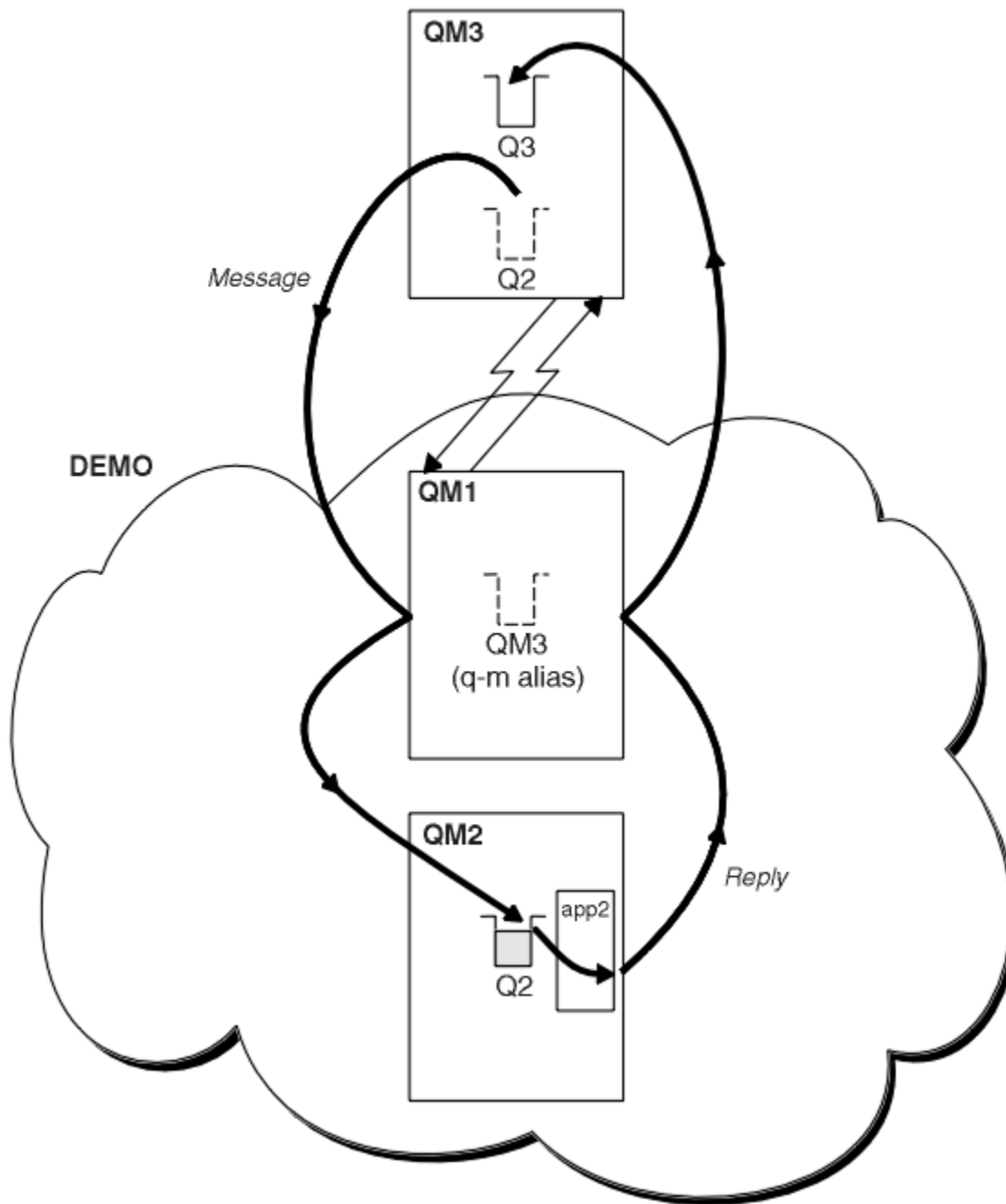


Figure 54. Putting from a queue manager outside the cluster

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 55 on page 394 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.

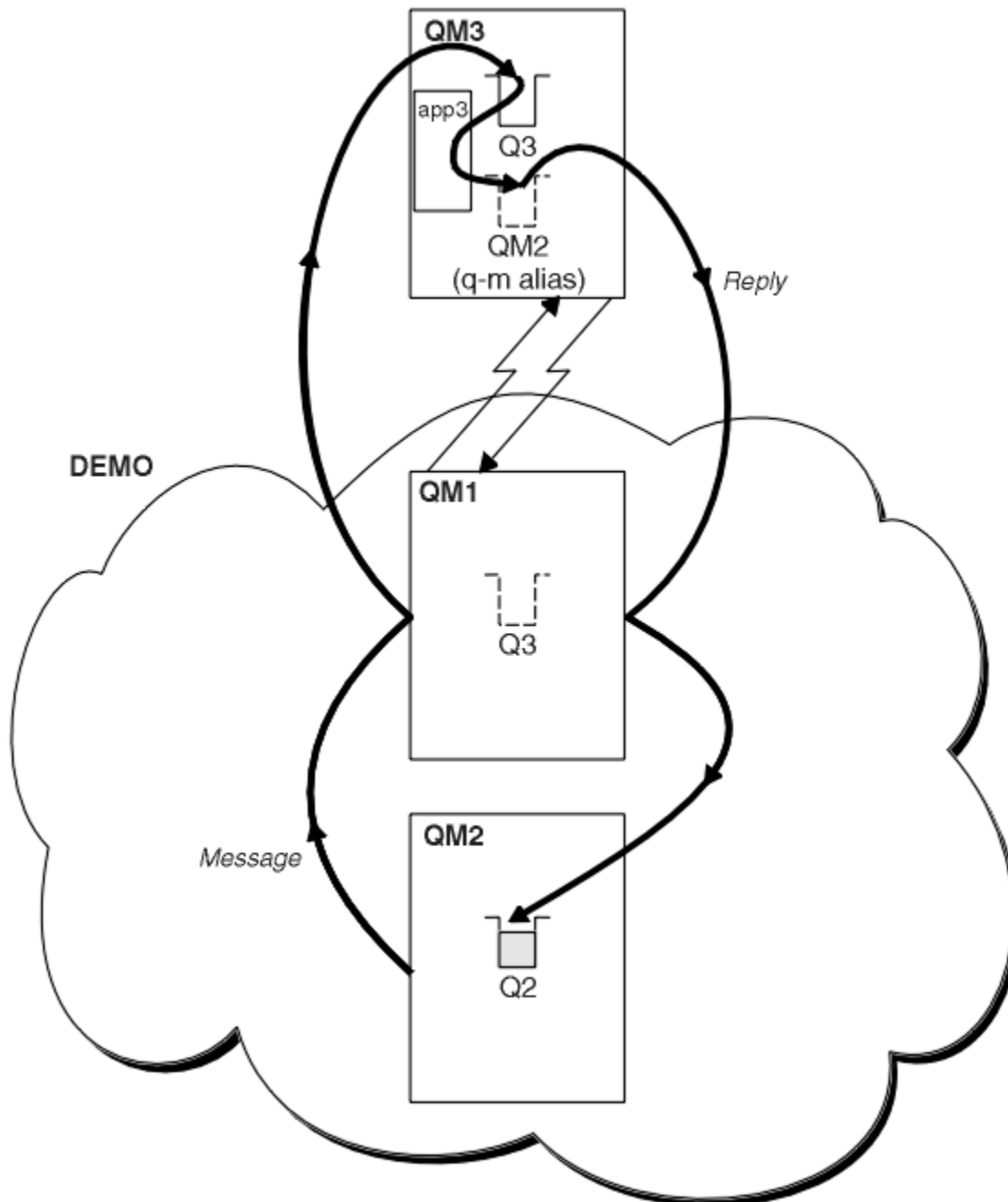


Figure 55. Putting to a queue manager outside the cluster

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 55 on page 394](#).

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 53 on page 389](#) in “[Configuring request/reply to a cluster](#)” on [page 388](#).

About this task

In this scenario, the queue manager outside the cluster, QM3 in [Figure 56 on page 397](#), 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 56 on page 397](#).

The procedure extends the example in [Figure 53 on page 389](#) in “[Configuring request/reply to a cluster](#)” on [page 388](#).

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

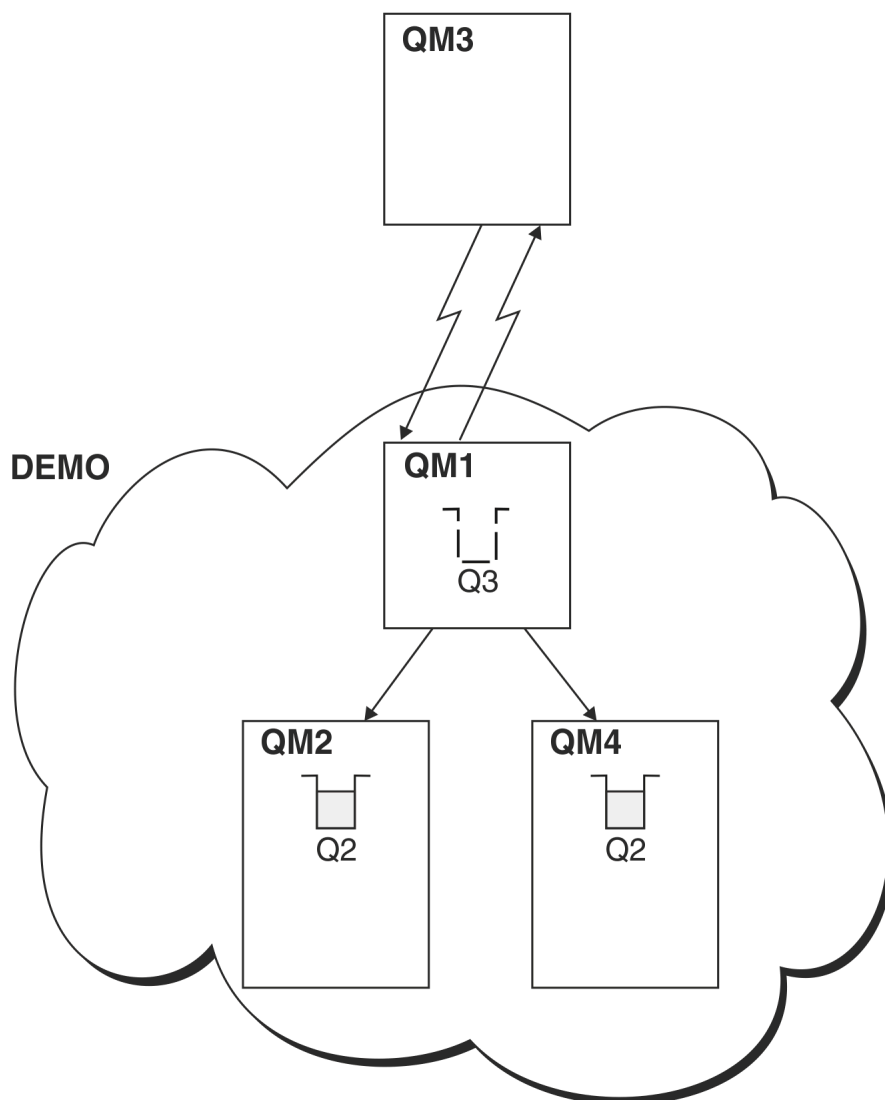


Figure 56. Putting from a queue manager outside the cluster

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.

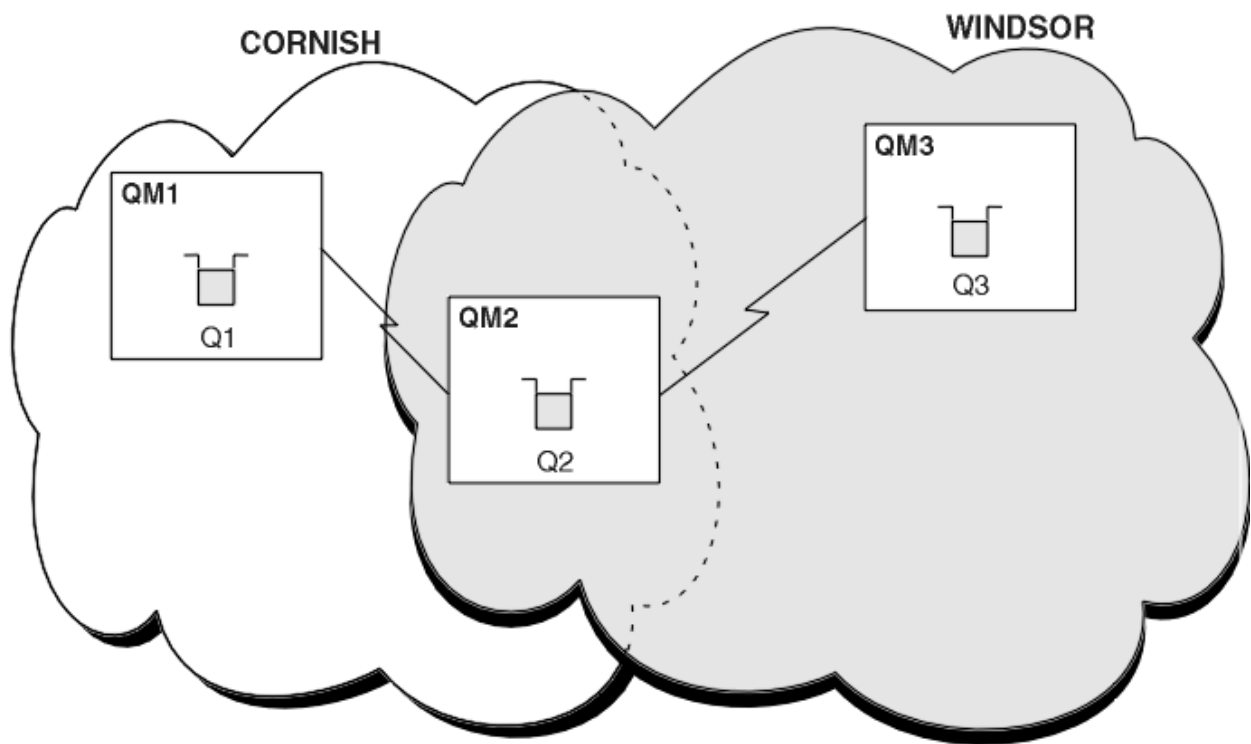


Figure 57. Bridging across clusters

Figure 57 on page 398 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 NAMLIST(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 57 on page 398, 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 355, 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 gateway queue manager QM1. QM3 routes the reply message to gateway queue manager QM1. QM1 routes the message to QM2.

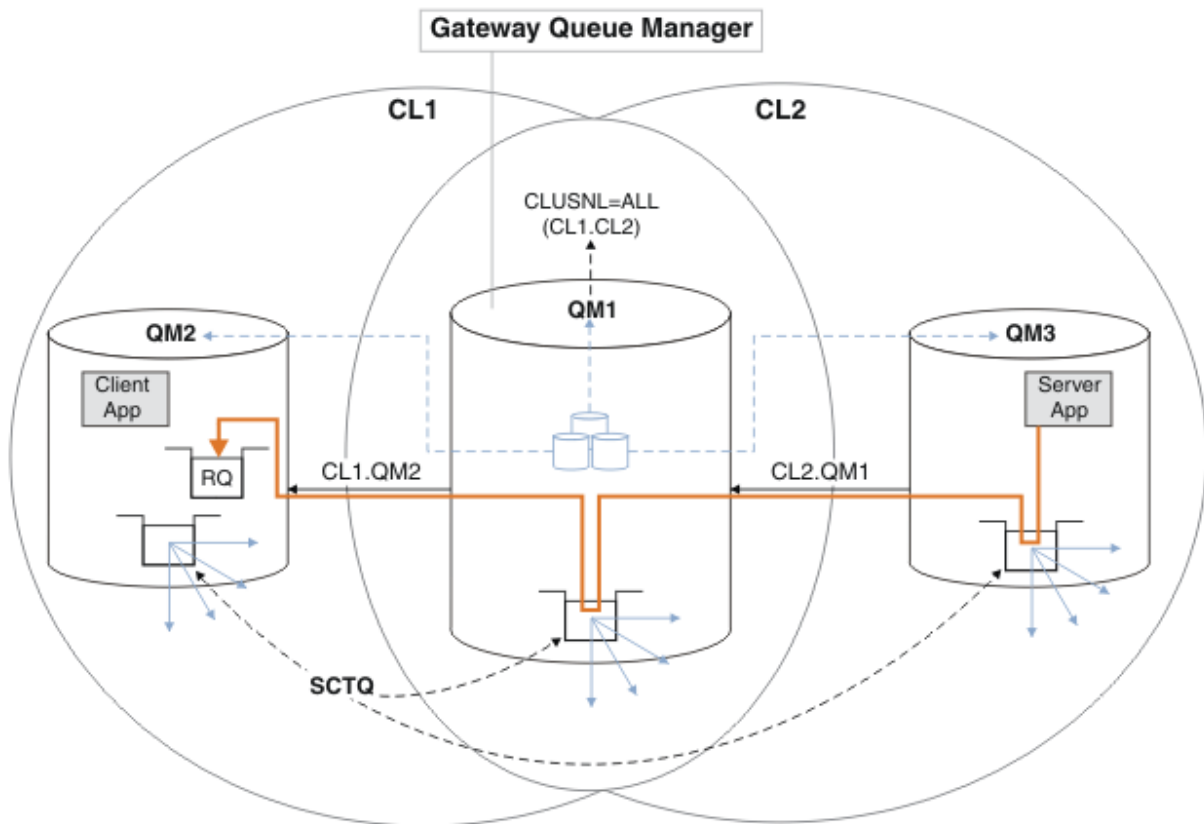


Figure 58. Using a queue manager alias to return the reply message to a different cluster

The way the routing works is as follows. Every queue manager in each cluster has a queue manager alias definition on gateway queue manager 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 gateway queue manager 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.

Gateway queue manager QM1 places the message on its cluster transmission queue for transfer to QM2. Gateway queue manager 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 gateway queue manager QM1, because both QM1 and QM2 are in the same cluster, CL1. Gateway queue manager 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 396](#)

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'
ReplyToQMgt=' '
```

- 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 requested 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.

 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](#).



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 387](#)

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 309](#)

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 322](#)

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 396](#)

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 59 on page 406 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 of 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.

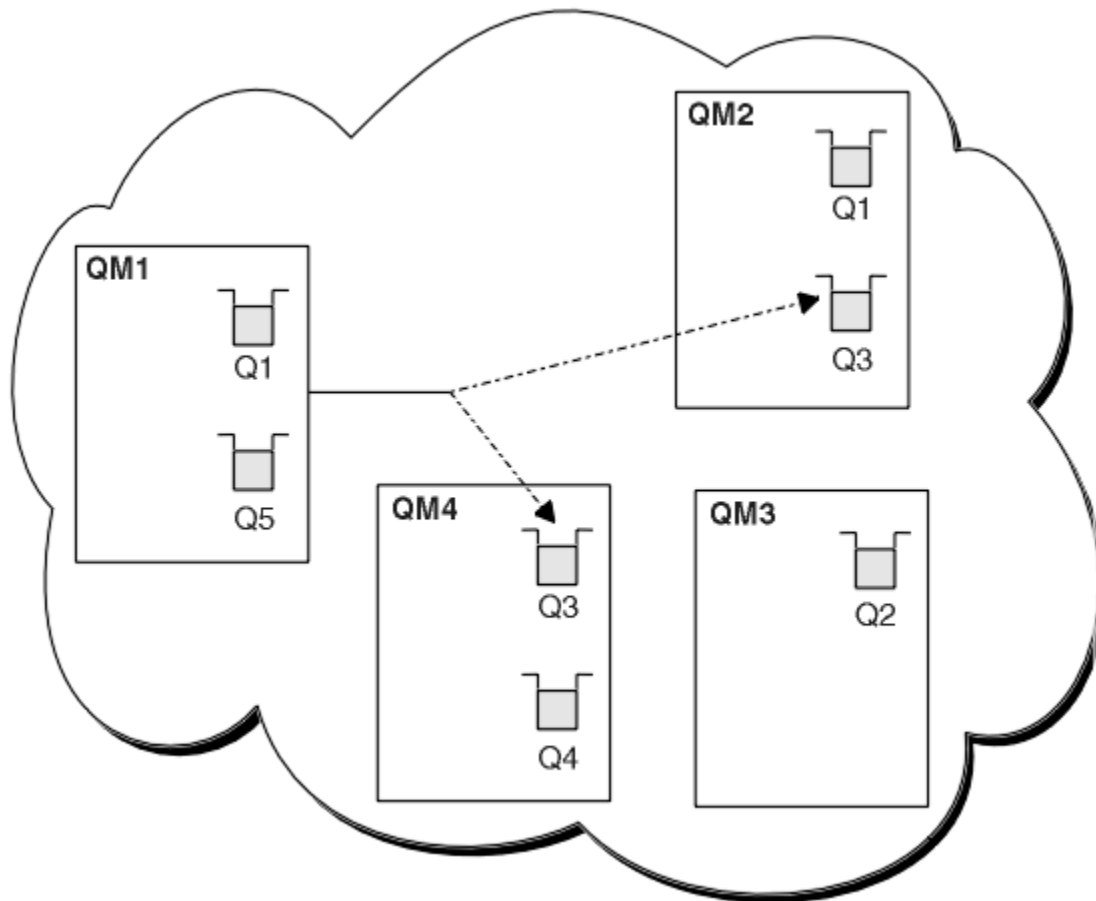


Figure 59. A cluster with multiple instances of the same queue

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)
```

z/OS 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 60 on page 408](#) shows the cluster set up by this task.

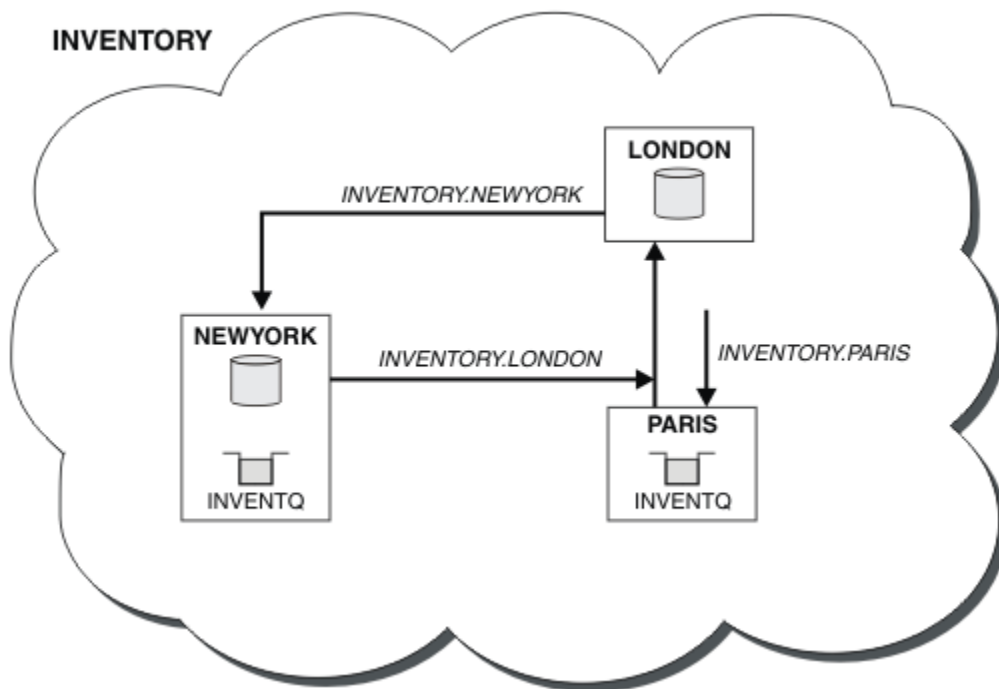


Figure 60. The INVENTORY cluster, with three queue managers

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')
```

 **z/OS** 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 61 on page 410 shows the cluster set up by this task.

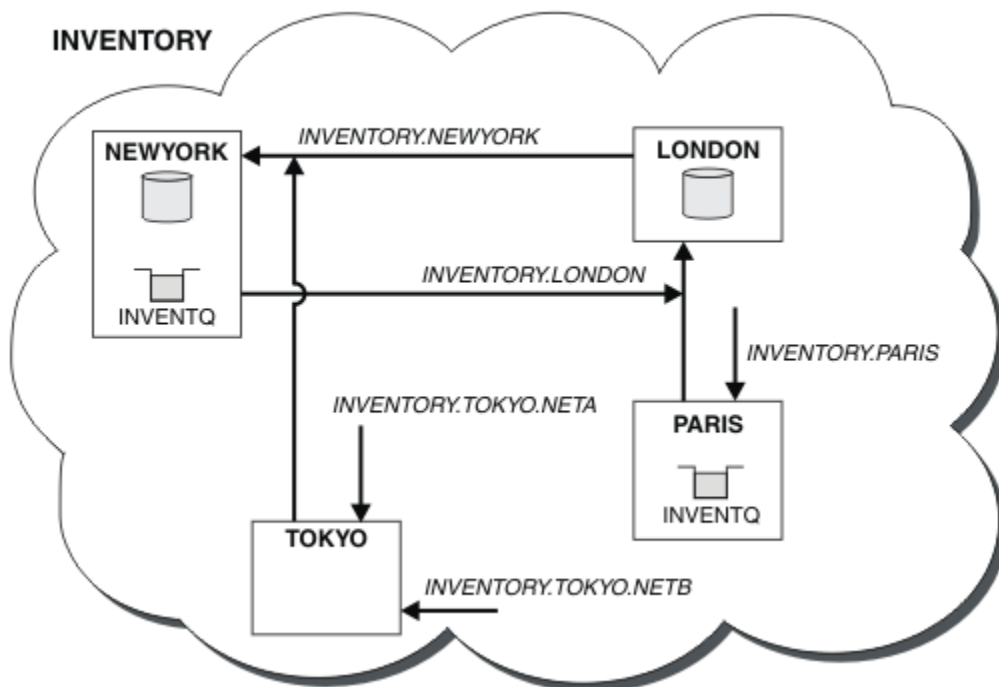


Figure 61. The INVENTORY cluster, with four queue managers

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 333

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 409. 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 333. 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 62 on page 414](#) shows the cluster set up by this task.

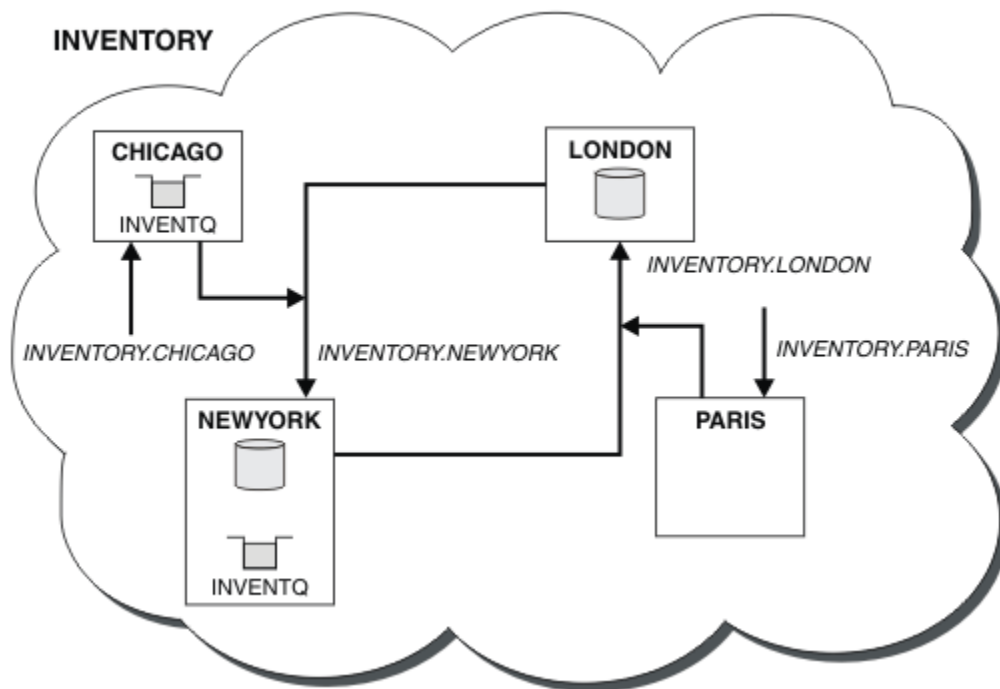


Figure 62. The INVENTORY cluster, with four queue managers

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 preferable 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 back up.

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. Start a listener program.

z/OS 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 63 on page 416 shows the cluster set up by this task.

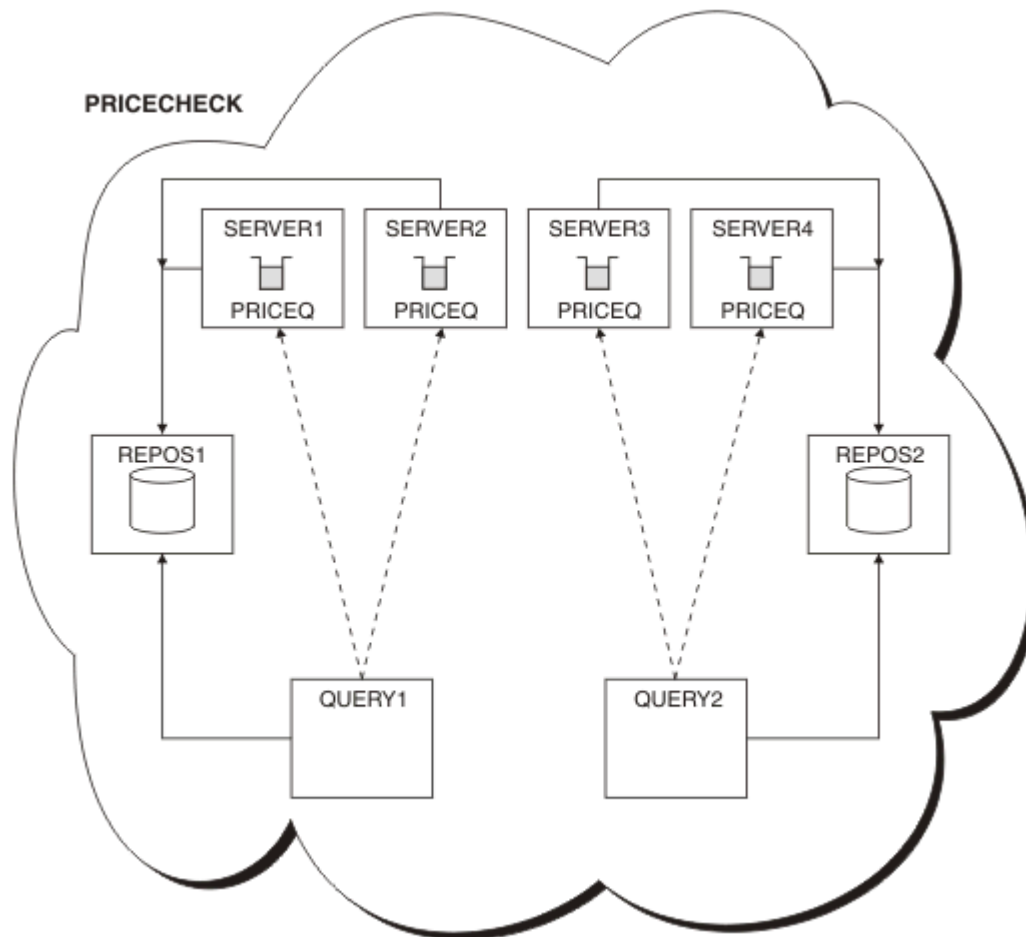


Figure 63. The PRICECHECK cluster, with four server queue managers, two repositories, and two query queue managers

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 333. 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 417](#) shows the cluster set up by this task.

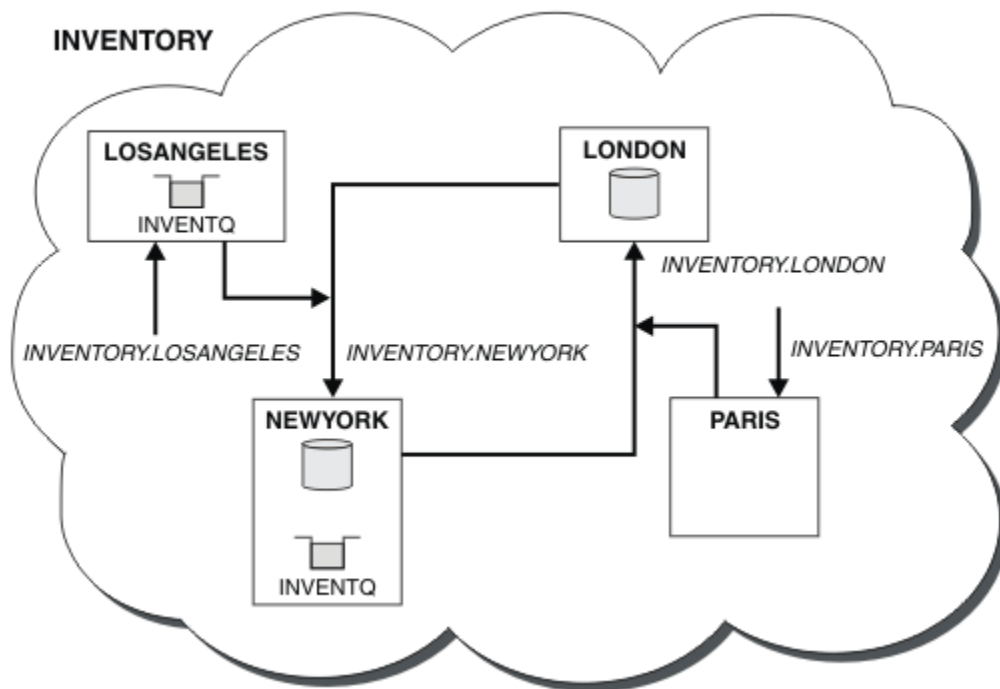


Figure 64. The INVENTORY cluster with four queue managers

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.

MQ00_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 MQ00_BIND_ON_GROUP, a route is selected when the queue is opened.

For MQ00_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 MQ00_BIND_ON_GROUP is specified but the messages are not grouped, the behavior is equivalent to MQ00_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 MQ00_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 MQ00_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.

Multi

Configuring a uniform cluster

Uniform clusters permit applications to be designed for scale and availability, and can connect to any of the queue managers within that uniform cluster.

Before you begin

For an introduction to clustering, see [Clusters](#). For an introduction to uniform clusters, see [“About uniform clusters”](#) on page 424.

About this task

Uniform clusters use IBM MQ clustering for communication between queue managers, and balancing of workload between queues. However they differ from typical IBM MQ clusters in the following ways:

- Uniform clusters typically have a smaller number of queue managers in the cluster. You should not create a uniform cluster with greater than 10 queue managers.
- Every member of the cluster has near-identical configuration.
- The cluster is typically used by a single application, or group of related applications.
- The number of application instances connecting to the cluster should be greater than, or equal to, the number of queue managers.

You can simplify both creating a uniform cluster, and subsequently keeping the configuration between the uniform cluster members identical, by using the automatic configuration and automatic clustering support.

Procedure

- [Learn about uniform clusters](#)

- [Create a uniform cluster](#)
- [Create a uniform cluster](#)
- [Suspend a queue manager from a uniform cluster](#)

Multi **About uniform clusters**

The objective of a uniform cluster deployment is that applications can be designed for scale and availability, and can connect to any of the queue managers within the uniform cluster. This removes any dependency on a specific queue manager, resulting in better availability and workload balancing of messaging traffic.  Uniform clusters are not available on IBM MQ for z/OS; queue sharing groups provide many of the capabilities of a uniform cluster.

Uniform clusters are a specific pattern of an IBM MQ cluster that provides a highly available and horizontally scaled small collection of queue managers. These queue managers are configured almost identically, so that an application can interact with them as a single group. This makes it easier to ensure each queue manager in the cluster is being used, by automatically ensuring application instances are spread evenly across the queue managers.

Uniform clusters remove some of the manual steps an administrator has to go through to create and administer a group of independent, interconnected queue managers. They move some client connection logic from the client to the queue manager, where information about levels of application activity can inform decisions at the clients, as to which queue managers they should connect to.

You can simplify both initial creation of a uniform cluster, and subsequently keeping the configuration between the uniform cluster members identical, by using the automatic configuration and automatic clustering support. When using this capability, one configuration file describes the cluster and another represents the MQSC configuration to apply to all queue managers in the uniform cluster. On each queue manager restart, the configuration is reapplied and the cluster automatically forms. See [“Creating a uniform cluster”](#) on page 438 for more details on using this feature.

To take full advantage of a uniform cluster, each application should also be scaled into multiple matching instances, preferably with at least as many instances as there are queue managers, if not many more.

An IBM MQ cluster, of whatever size, provides multiple capabilities:

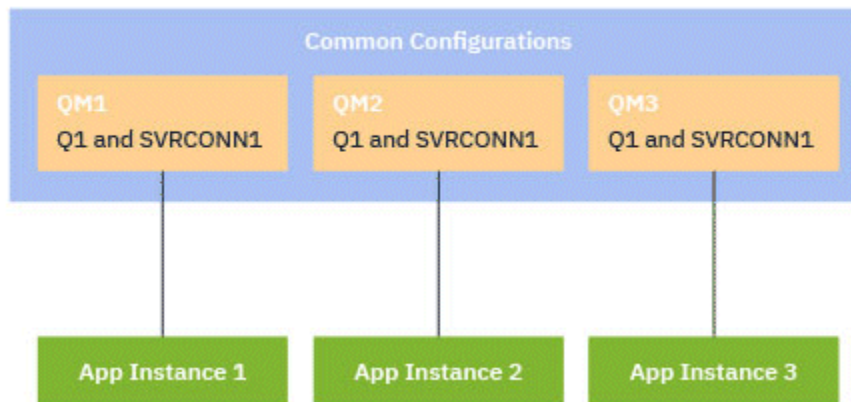
- A directory of all clustering resources, discoverable by any member in a cluster
- Automatic channel creation and connectivity
- Horizontal scaling across multiple matching queues, using message workload balancing
- Dynamic message routing, based on availability

Uniform clusters use IBM MQ clustering for communication between queue managers, and balancing of workload between queues. However they differ from typical IBM MQ clusters in the following ways:

- Uniform clusters typically have a smaller number of queue managers in the cluster. You should not create a uniform cluster with greater than 10 queue managers.
- Every member of the cluster has near-identical configuration.
- The cluster is typically used by a single application, or group of related applications.
- The number of application instances connecting to the cluster should be greater than, or equal to, the number of queue managers.

In a uniform cluster pattern, all queue managers in the cluster offer the same messaging services. For example, you might configure all cluster members to have the same local queues defined, and allow client applications to connect to any member of the cluster. You might also have the same server connection channels defined, and possibly the same authority records, channel authentication rules, and so on. However, members of the cluster can still have some differences in objects and configuration. For example, some applications might create temporary dynamic queues while they are connected to a queue manager. Also, some configuration updates might be rolled out across the members over a period of time; for example, new or updated certificates. As with regular IBM MQ clusters, two of the queue managers will require additional configuration to make them full repository queue managers.

The following diagram shows that the queue managers have similar configurations. They define the same queue called Q1 and the same server connection channel SVRCONN1.



Note that, for multiple queue managers with identical server connection channel names to work with a single client-channel definition table (CCDT), you must use the updated CCDT format introduced in IBM MQ 9.1.2. See [“Configuring a JSON format CCDT”](#) on page 45.

Application names and application instances

An application name is displayed as the `APPLTAG` attribute of the **DISPLAY CONN(*) TYPE CONN** command. From IBM MQ 9.1.2, there is a change to the way in which the application name is set.

An instance of an application is a set of closely related connections that provide one *unit of execution* for that application. Typically, this is a single operating system process, which can have a number of threads and associated IBM MQ connections.

For more information about application name and application instances, see [Application development concepts](#).

Reconnectable clients

Reconnectable clients can be moved in order to achieve an even workload distribution whereas, by definition, a non-reconnectable client cannot be reconnected to a different queue manager.. However, there might still be a good reason to connect a non-reconnectable client into a uniform cluster: For example, because the client creates some form of persistent state, and some other mechanism is used to ensure that there are instances of the application running in each of the queue managers.

Locally bound applications

Uniform clusters are expected to have IBM MQ applications connecting as client applications, rather than locally bound applications. Locally bound applications are not prevented from connecting to uniform cluster members, but uniform clusters cannot achieve even workload distribution with locally bound applications, because they cannot connect to any other member of the cluster.

Related tasks

[Specifying the application name in supported programming languages](#)

► Multi **Automatic application balancing**

Automatic application balancing greatly enhances application distribution and availability by enabling an IBM MQ uniform cluster to closely manage application distribution across the cluster, rather than relying on randomization, or a manual pinning of applications to specific queue managers.

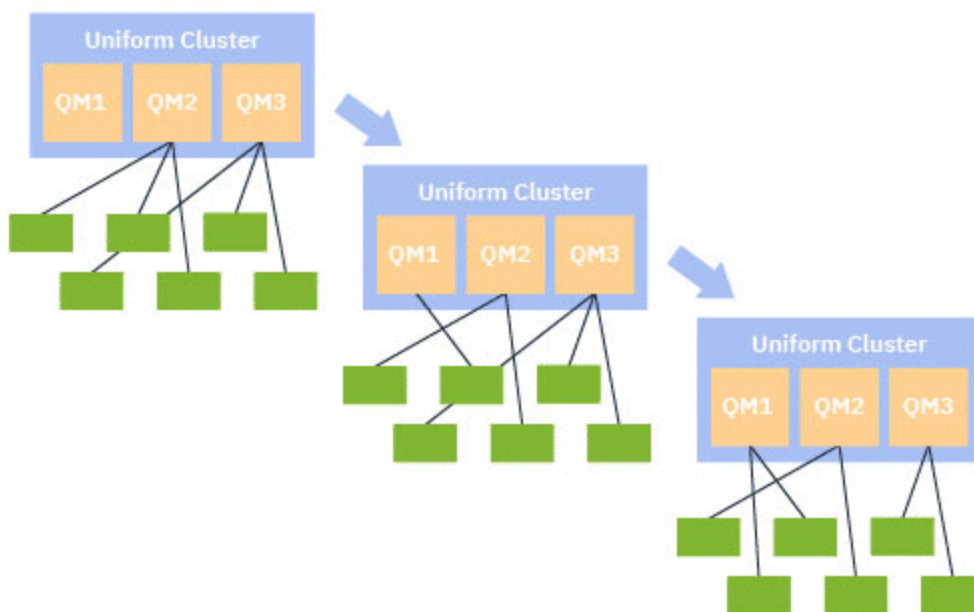
Automatic balancing across a set of clustered queue managers is supported for applications written in C, JMS, IBM MQ .NET, XMS .NET.

When there are at least as many instances of the same application as there are queue managers, the uniform cluster constantly ensures that every queue manager has at least one instance of the application connected.

Applications can remove a specific affinity to a queue manager, and instead use a client channel definition table (CCDT) to randomize connectivity to the group of queue managers in the uniform cluster safely. Applications can do this for the following reasons:

- When there are enough consuming application instances, there is always an instance of the application processing messages.
- When you stop a queue manager, any connected application instances are evenly distributed across the remaining queue managers in the cluster.
- When you start a queue manager, any application instances connected to other queue managers in the cluster are automatically rebalanced to include the newly started queue manager.

This means that the uniform cluster continually ensures that applications are optimally distributed, maximizing message processing, even in the event of planned and unplanned outages.



To achieve auto-balancing, the queue managers in the uniform cluster periodically share information between themselves. They do this by publishing metadata on system topics under the reserved `$SYS/MQ` branch of the topic tree. Every queue manager in the uniform cluster subscribes to messages published by other queue managers, and builds a picture of the state of the applications in the uniform cluster.

The queue managers monitor the distribution of client applications across the entire cluster. When the number of applications connected to a specific queue manager is sufficiently low that it determines the cluster is unbalanced, that queue manager publishes a request on a system topic to one of the other queue managers in the cluster.

When the message is received, the target queue manager requests one of its client applications to redirect to the requesting queue manager. The client application receives the redirect request, closes its connection, and reconnects to the requesting queue manager. This automatic balancing mechanism is transparent to the application. For more information, see [“How automatic balancing works”](#) on page 427.

By periodically distributing meta-data about connected applications, the uniform cluster can achieve a broadly balanced ratio of client applications to queue managers over time. To prevent rapid successive redirect events from occurring, the auto-balancing algorithm limits the rate at which redirect requests are made.

You can monitor the current state of applications across queue managers in a cluster, and monitor application instances. For more information, see [Monitoring application balancing](#). You can also solve various issues with application balancing, as described in [Application balancing troubleshooting](#).

Rebalancing is only useful for applications with a long connection time. If you have client applications with short connection times, for example client applications that are written to connect and disconnect regularly to different queue managers, you should configure them as not being reconnectable. This removes them from the set of applications that the queue managers try to balance.

Related concepts

[“How automatic balancing uses automatic reconnect ” on page 429](#)

Uniform cluster automatic balancing makes use of enhancements to the existing automatic reconnect feature of IBM MQ.

How automatic balancing works

In the uniform cluster, client connections are grouped together based on the application name. Applications that connect to any member of the uniform cluster using the same application name are considered to be equivalent to any other applications using the same application name.

Auto-balancing ensures an even spread of application instances across members of the cluster; see [“Application names and application instances” on page 425](#) for more information. Use the `DISPLAY APSTATUS` command to display the status of one or more applications, and application instances, connected to a queue manager or a uniform cluster.

For example, you could set all instances of an insurance request application to have an application name of "INSURANCE.REQUESTS". Related connections from this application will be automatically grouped into instances where appropriate, with all balancing performed on a per-instance basis.

When new instances of the application connect to a member of the uniform cluster, the auto-balancing algorithm assesses which queue managers have the fewest instances of INSURANCE.REQUESTS and redirects some connections to those queue managers.

Automatic balancing is enabled in the following circumstances only:

- The SHARECNV value of the channel is greater than zero.
- One of the following is true:
 - The client application specifies MQCNO_RECONNECT
 - The `mqcClient.ini` file specifies **Defrecon**=YES

Note: Applications with queue manager affinity - for example because of a durable subscription or dynamic reply to queue - cannot safely be rebalanced and should either use MQCNO_RECONNECT_QMGR or no reconnect option at all.

When a client is redirected to an alternative queue manager it will, as usual, use the local client channel definition tables (CCDTs) to locate the connection information for the new target. It is, therefore, important for smooth and efficient operation of auto-balancing, that clients use a CCDT containing an entry for each member of the uniform cluster, as well as any queue manager group used to balance the initial connections.

Use of a JSON format CCDT simplifies this, as it allows multiple connections using the same server connection name. For more information, see [“Configuring a JSON format CCDT” on page 45](#).

Related concepts

[“How automatic balancing uses automatic reconnect ” on page 429](#)

Uniform cluster automatic balancing makes use of enhancements to the existing automatic reconnect feature of IBM MQ.

Automatic balancing of JMS applications

When [Jakarta Messaging 3.0](#) or [Java Message Service 2.0](#) applications are balanced automatically, the underlying groups of IBM MQ connections that JMS applications create are moved together.

From IBM MQ 9.3.0, the **dynamicallyBalanced** property is available when configuring `ActivationSpecs`. This property specifies whether an MDB can be requested to receive messages from a different queue manager as part of application balancing in a uniform cluster. For more information, see [Configuring the resource adapter for inbound communication](#).

For handling JMS connections, uniform clusters have the concept of an *application instance*. For JMS, an *application instance* is defined as a JMS Connection and all its associated JMS Sessions. v 10.0.0 From IBM MQ 10.0.0, the previous definition is the default but you can instead specify that an application instance is defined as all the JMS Connections from a single Java virtual machine, as described in [Application Instances in IBM MQ classes for JMS/Jakarta Messaging](#).

A unique connection tag is allocated on the client connection that corresponds to the JMS Connection, and the same tag is then applied to the client connections that correspond to JMS Sessions that are created by that JMS Connection. v 10.0.0 (Or from IBM MQ 10.0.0, the same tag is applied to all the JMS connections for a single Java virtual machine, if you specified the JVM mode of operation.)

For example, if a pair of client applications are running JMS applications against a uniform cluster with a single active queue manager (Queue Manager 1):

- Client 1 creates a connection factory on which it sets an application name of "App1", and creates a JMS Connection and three JMS Sessions. Client 1 creates four client connections in Queue Manager 1 each sharing the same connection tag, and this is treated as a single instance of "App1".
- Client 2 also creates a connection factory on which it sets an application name of "App1", and creates a JMS Connection and two JMS Sessions. Client 2 creates three client connections each sharing the same connection tag (distinct from that assigned to Client 1), and this is treated as a single, separate, instance of "App1".
- So the queue manager sees two instances of "App1"

When automatic balancing is performed, application instances are moved. A queue manager chooses an application instance (a group of client connections sharing the same connection tag) and requests that the instance moves to a different queue manager. The client code receives the request and ensures that all the related connections (corresponding to a JMS Connection and its associated JMS Sessions) move to the new queue manager.

For example, take the set of application instances previously outlined, and assume that a new queue manager (Queue Manager 2) starts up in the uniform cluster.

Queue Manager 2 has no work but Queue Manager 1 has 2 instances of "App1", so Queue Manager 2 requests that Queue Manager 1 transfer an instance of "App1" to Queue Manager 2.

Queue Manager 1 chooses an instance of "App1" to be moved. For the purposes of the example, assume that it chooses the instance created by Client 1.

- Queue Manager 1 sends a request to Client 1 to move its instance of "App1" to QM2.
- The client closes its four existing client connections to Queue Manager 1 and creates four new connections to Queue Manager 2.
- The JMS Connection and its JMS Sessions, except for a short pause in processing, should not normally be disturbed.

Note:

An application might receive a JMS exception if certain operations are in progress at the time that an application instance is moved.

The JMS exception will have a linked IBM MQ exception, from which the reason code can be retrieved to determine the cause of the failure.

The expected reason codes are as follows:

MQRC_CALL_INTERRUPTED

This occurs when, for example, a message which is persistent (the default in JMS) is put outside a syncpoint, but the operation is interrupted by a reconnection.

MQRC_BACKED_OUT

This occurs when, for example, an attempt to put a message inside a syncpoint is interrupted by a reconnection.

Related concepts

[“How automatic balancing works” on page 427](#)

In the uniform cluster, client connections are grouped together based on the application name. Applications that connect to any member of the uniform cluster using the same application name are considered to be equivalent to any other applications using the same application name.

[“How automatic balancing uses automatic reconnect ” on page 429](#)

Uniform cluster automatic balancing makes use of enhancements to the existing automatic reconnect feature of IBM MQ.

How automatic balancing uses automatic reconnect

Uniform cluster automatic balancing makes use of enhancements to the existing automatic reconnect feature of IBM MQ.

In versions of IBM MQ before IBM MQ 9.2.0, the automatic reconnect feature automatically reconnects to a standby instance of a queue manager, or to a different queue manager, based on the connection details provided, typically a connection name list, or a Client Channel Definition Table (CCDT).

In some circumstances the IBM MQ client silently performs the reconnection without the application being aware it has happened. The decision of which queue manager to reconnect to is entirely down to the sequence of connection names in a connection name list, or the workload balancing configuration in the CCDT.

From IBM MQ 9.2.0 a reconnect request can be sent to a client containing a hint of which queue manager the client should reconnect to. In many reconnect scenarios, such as a queue manager failure, or the administrator issuing the command **endmqm -r**, a queue manager name is not included in the hint information, and automatic reconnect behavior works as it did before IBM MQ 9.2.0.

However, if you have configured a uniform cluster, automatic application balancing periodically sends reconnect requests to clients, in order to achieve a balanced cluster. In these cases the uniform cluster specifies a queue manager name in the reconnection hint to ensure that client connections are moved to the queue managers that have the least connections.

For automatic balancing to work the following settings must be in place:

- IBM MQ applications use CCDTs to retrieve connection information from.
- CCDTs contain an entry for every queue manager in the uniform cluster.

If this is not the case, the cluster cannot automatically balance applications across all members of the cluster.

If an application is using a version of the IBM MQ client that is earlier than IBM MQ 9.2.0, and it is configured to support automatic client reconnection, then it might be sent a request by the uniform cluster to go through its reconnect steps.

The client will not be asked to reconnect to a specific queue manager, but instead goes through the same sequence of reconnect logic it would do for other reconnect events. To achieve even distribution of client applications prior to IBM MQ 9.2.0 across the uniform cluster, ensure that clients are configured to use CCDTs containing evenly weighted entries for each member of the cluster.

Applications can take several reconnect attempts before connecting to a queue manager that needs the additional instance, so it is a less efficient way of achieving even distribution of applications across the cluster. Automatic balancing can take longer to achieve in these environments.

IBM MQ clients not supporting automatic client reconnection

If an application is using a version of the IBM MQ client that does not support automatic client reconnection, the application might receive a failure return code from an MQI call.

If your application has not been designed to handle failures and perform re-connections manually, it might be necessary to disable auto balancing for those applications.

Note: Auto balancing is enabled for any application that is identified as being reconnectable, that is, the application has MQCNO_RECONNECT in its effective connection options.

Related tasks

[“Creating a new uniform cluster” on page 438](#)
How you create a new uniform cluster.

Influencing application re-balancing in uniform clusters

With automatic application balancing (a feature of uniform clusters), an application connection can be asked to move to an alternative queue manager at any point in its lifecycle.

Introduction

From IBM MQ 9.3.0, the balancing algorithm automatically tries to take into account the state of applications to minimize disruption to application flow. This can be tuned to suit particular applications or application instances by giving IBM MQ additional information about the type of application, or pattern of IBM MQ activity performed by this application.

Usually, the person developing or deploying a client application is likely to be best placed to understand this pattern and supply this information to the queue manager, (see [Deploying flexible and scalable client applications](#)) but it might also, or additionally, be tuned by an administrator.

Note that if the queue manager is unable to achieve an even distribution of applications within a reasonable period of time, application connections might still be rebalanced to other queue managers without waiting for a convenient time in their IBM MQ flow.

This can also be tuned to meet requirements. If it is more important to quickly achieve an even distribution of applications, you can configure the product to wait less time to find a suitable time to rebalance an application. Alternatively, if it is more important to prevent disruption to applications, it is possible to configure the product to always wait for a convenient time to move the application.

See [Deploying flexible and scalable client applications](#) for further overview information.

For .NET applications, see [“Influencing application re-balancing in .NET” on page 433](#) for further information.

For .XMS.NET applications, see [Properties of ConnectionFactory](#) for more information.

For JMS Applications, see [“Influencing application re-balancing in IBM MQ classes for JMS” on page 434](#) for further information.

Default application balancing behavior

By default, the transaction/unit of work state of an applications interaction with a queue manager is considered for all applications.

For local transactions, automatic application balancing avoids issuing re-balancing requests to applications that are currently involved in a transaction. While this does not eliminate the possibility of an application receiving a backed out return code, as reaching the configured re-balancing timeout or a genuine outage could still cause such a return code, it does mean that applications will not usually be asked to reconnect while they are in the middle of a transaction.

For applications that begin a new transaction almost immediately after completing the previous one, there might be a delay for the initial call in the new transaction while re-balancing completes. This ensures that automatic application balancing is still able to achieve an even distribution of applications between the queue managers in a uniform cluster.

If you have applications that use longer running transactions you might want to consider increasing the value of the re-balancing timeout, or disable this constraint entirely. See [“Configuring the balancing behavior” on page 431](#) for links on how to control this in the MQI and .NET, or 'Designing client applications for fault tolerance and scalability' for the code level equivalent.

Request-reply balancing

When the application type is specified as **Request-Reply**, one response GET is expected for every PUT the application instance performs. If the application instance involves multiple threads, or deals with requests and responses in batches, multiple requests and responses can be in flight at any given time.

The application is not considered eligible to move, until the number of requests sent is equal to the number of responses received, or the backstop value of the timeout is exceeded.

An exception to this is when message expiry is configured for a request message. It is assumed that responses should be received within the expiry interval of the request message, and when all request messages have expired, the balancing algorithm no longer waits for additional responses before considering the instance eligible to move.

If multiple requests are outstanding, only the latest expiry among the request messages sent is considered. When meaningful expiry values are in use, you should configure the **Timeout** balancing parameter for the application to be at least as high a value as any sent message expiry, to avoid cutting short any expected request/response expiry window.

The preceding pattern is only suitable for applications that expect to have periods in which there are no outstanding requests. Complex multi-threaded applications, which constantly send and receive messages, for example, might never become eligible to rebalance under this pattern.

Notes:

- No attempt is made to correlate specific requests and responses so, if an earlier response within a batch of in flight messages expires, the application might still wait until the latest request expires before being eligible to balance.
- In particular, care is needed if combining an unlimited expiration time and expiring messages, for similar reasons.

If request messages with a limited expiry are outstanding, and new messages are sent with an unlimited expiry time, the unlimited expiry timeout is *not* taken into account by the balancing algorithm, which continues to honor the current latest expiry time.

Otherwise, earlier responses having expired could prevent the application from ever being eligible to move. Correspondingly, if unlimited expiry timeout replies are outstanding, but expiring requests are subsequently sent, the wait time is reduced to the longest (limited) expiry.

In general, you should avoid a single application instance sending both expiring and non-expiring request messages in a balanced application, as eligibility to rebalance becomes harder for a developer or administrator to accurately track or define.

- Only the expiry time specified by the sending application (for example, in the MQI the value of **MQMD.Expiry**) is considered when determining how long to wait for replies. Subsequent modifications to this value, for example, use of CAPEXPY will not effect the wait time.

Configuring the balancing behavior

To influence precisely when IBM MQ re-balances applications, certain client application environments can provide information at connect time about the messaging pattern being employed.

This information is provided in a new structure referred to as *Balancing Options*.

For the MQI, see [“Configuring balancing behavior using the MQI”](#) on page 432.

For the .NET client equivalent of this structure, see [“Influencing application re-balancing in .NET”](#) on page 433.

For the JMS approach to setting these options, see [“Influencing application re-balancing in IBM MQ classes for JMS”](#) on page 434 for further information.

Other client environments do not currently support supplying this structure at connect time.

Configuring balancing behavior using the MQI

To influence precisely when IBM MQ re-balances applications, certain client application environments can provide information at connect time about the messaging pattern being employed.

In the MQI, the balancing options structure is known as the [MQBNO](#).

If no *Balancing Options* are provided in your program, supporting clients derive this information in the [Application stanza](#) or [ApplicationDefaults stanza](#) in the `client.ini` file deployed alongside the client application.

Note: These stanzas are identical, except the *Application* version contains a **Name** field to identify which application these options apply to.

If either form of stanza is supplied, all fields are required to be present except **BalanceOptions** which is assumed to be none if not explicitly set.

The order of preference for supplying options is:

1. An MQBNO structure is supplied by the application on CONNX and used in entirety
2. Or, the matching named *Application* stanza, if present, is solely used to generate one
3. Or, the *ApplicationDefaults* stanza, if present, is solely used to generate one
4. Or, no MQBNO flows for this connection.

You can supply three key pieces of information from either the MQBNO structure or the `client.ini` file:

1. The **ApplicationType** or pattern of application.

This field indicates to IBM MQ the general pattern of IBM MQ activity in which this application participates.

Three types of application are supported:

Simple

No specific rules should be applied beyond the defaults described in [“Default application balancing behavior”](#) on page 430 .

Request-Reply

After each MQPUT call, a matching MQGET call is expected for a response message. See [“Request-reply balancing”](#) on page 431 for more details.

Managed client

Re-balancing requests are always dispatched immediately to the client, which re-balances at a point it considers appropriate, for example, the JEE resource adapter would register in this manner.

2. The **Timeout** after which re-balancing can interrupt application activity
3. Specific **BalanceOptions**

Examples of when your application might be re-balanced

Example 1

You have written an application that puts messages under syncpoint and commits the batch of messages by issuing an MQCMIT call. When the MQCMIT call completes, the application starts putting messages under a new syncpoint.

Suggested IBM MQ configuration

Default options sufficient

Result

An application instance is moved after an MQCMIT call succeeds (or fails), once the configured number of transactions has been met.

By default, if a batch of messages exceeds 10 seconds, it might be rolled back if a rebalance has been requested. If you expect transactions to regularly exceed this limit and require this to be permitted, you can extend the **Timeout** appropriately.

Example 2

You have written an application that puts one message to a cluster queue instance, and another application replies to a local temporary dynamic queue with a message, after processing the request. When the request has been destructively read from the local queue the application puts its next request message.

Suggested IBM MQ configuration

Set Type to MQBNO_BALTYPE_REQREP

Result

An application instance is moved when an application completes an MQGET call, at which point the application instance moves to another queue manager. Any subsequent MQPUT calls are carried out on the new queue manager.

MQBNO

ApplicationType

Influencing application re-balancing in .NET

From IBM MQ 9.3.0, additional constants are available for you to set the balancing option properties using a hash table from the application when you use the MQQueueManager class to connect to the queue manager.

The following constants are the ones you use to influence application balancing in .NET:

Re-balancing application type

The type of balancing action; represented by the constant

MQC.BALANCING_APPLICATION_TYPE_PROPERTY

- You must use this property to set the **ApplicationType** field of the MQBNO structure.

You must set values of type integer, and the possible values are:

MQC.BALANCING_APPLICATION_TYPE_SIMPLE

Simple balancing; no specific rules are applied in addition to those described in [“Influencing application re-balancing in uniform clusters”](#) on page 430. This is the default value.

MQC.BALANCING_APPLICATION_TYPE_REQUEST_REPLY

Request-Reply balancing; after each **MQPUT** call, a matching **MQGET** call is expected for a response message. Balancing is delayed until such a message is received, or the request message **EXPIRY** has been exceeded.

If the reconnection is enabled by the application and this property is not set,

MQC.BALANCING_APPLICATION_TYPE_SIMPLE is used

Re-balancing options

The balancing options set by the issuing application; represented by the constant

MQC.BALANCING_OPTIONS_PROPERTY

- You must use this property to set the **BalanceOptions** field of the MQBNO structure.

You must set values of type integer, and the possible values are:

MQC.BALANCING_OPTIONS_NONE

No options are set. This is the default value

MQC.BALANCING_OPTIONS_IGNORE_TRANSACTIONS

Setting this option allows applications to be rebalanced even if in the middle of a transaction.

If the reconnection is enabled by the application and this property is not set,

MQC.BALANCING_OPTIONS_NONE is used.

Re-balancing timeout

Timeout after which re-balancing might interrupt application activity; represented by the constant

MQC.BALANCING_TIMEOUT_PROPERTY

- You must use this property to set the [Timeout](#) field of the MQBNO structure.

You must set values of type integer, and the possible values are:

MQC.BALANCING_TIMEOUT_AS_DEFAULT

The set default timeout value. This is the default value

MQC.BALANCING_TIMEOUT_IMMEDIATE

Immediate timeout occurs

MQC.BALANCING_TIMEOUT_NEVER

No timeout occurs

Note: You must provide one value only from the defined values, or a value of 0-999999999 seconds.

Deploying flexible and scalable client applications

MQBNO

Influencing application re-balancing in IBM MQ classes for JMS

From IBM MQ 9.4.0, additional constants are available for you to set the balancing option properties on a **ConnectionFactory**. These constants are only applicable if the **WMQ_PROVIDER_VERSION** is set to 7. Request_reply applications in a uniform cluster must allow for the possibility of missed replies.

- [“The constants available”](#) on page 434.
- [“The potential for lost messages at balancing of REQUEST_REPLY applications”](#) on page 435.

The constants available

The following constants are the ones you use to influence application balancing in IBM MQ classes for JMS:

Re-balancing application type

The type of balancing action; represented by the constant

WMQConstants.WMQ_BALANCING_APPLICATION_TYPE

- You must use this property to set the [ApplicationType](#) field of the MQBNO structure.

You must set values of type integer. These are the possible values:

WMQConstants.WMQ_BALANCING_APPLICATION_TYPE_SIMPLE (default)

Simple balancing; no specific rules are applied in addition to those described in [“Influencing application re-balancing in uniform clusters”](#) on page 430.

WMQConstants.WMQ_BALANCING_APPLICATION_TYPE_REQUEST_REPLY

Request-Reply balancing; after each **MQPUT** call, a matching **MQGET** call is expected for a response message. Balancing is delayed until such a message is received, or the request message **EXPIRY** has been exceeded.

If reconnection is enabled by the application and this property is not set,

WMQConstants.WMQ_BALANCING_APPLICATION_TYPE_SIMPLE is used.

Re-balancing options

The balancing options set by the issuing application; represented by the constant

WMQConstants.WMQ_BALANCING_OPTIONS

- You must use this property to set the [BalanceOptions](#) field of the MQBNO structure.

You must set values of type integer. These are the possible values:

WMQConstants.WMQ_BALANCING_OPTIONS_NONE (default)

No options are set.

WMQConstants.WMQ_BALANCING_OPTIONS_IGNORE_TRANSACTIONS

Setting this option allows applications to be rebalanced even if in the middle of a transaction.

If reconnection is enabled by the application and this property is not set,

WMQConstants.WMQ_BALANCING_OPTIONS_NONE is used.

Re-balancing timeout

The timeout after which re-balancing might interrupt application activity; represented by the constant

WMQConstants.WMQ_BALANCING_TIMEOUT

- You must use this property to set the **Timeout** field of the MQBNO structure.

You must set values of type integer. These are the possible values:

WMQConstants.WMQ_BALANCING_TIMEOUT_AS_DEFAULT (default)

The set default timeout value. By default, this value is 10 seconds.

WMQConstants.WMQ_BALANCING_TIMEOUT_IMMEDIATE

Immediate timeout occurs.

WMQConstants.WMQ_BALANCING_TIMEOUT_NEVER

No timeout occurs.

a value between 1 and 999999999

This represents a value in seconds.

Note: You must provide one value only from the defined values, or a value of 0-999999999 seconds.

These properties are also settable in the JNDI representations of Connection Factories using either the JMSAdmin or IBM MQ Explorer interfaces.

The potential for lost messages at balancing of REQUEST_REPLY applications

In IBM MQ classes for JMS (and IBM MQ classes for Jakarta Messaging), request/reply functionality is implemented by setting the **JMSReplyTo** property on the request message, which is used by the responding application to determine whether the reply is sent. In JMS terms the **JMSReplyTo** property is a **Destination**.

When this is translated into IBM MQ operations, the **JMSReplyTo** property is sent as a fully-qualified queue URI - identifying a queue on a specific queue manager.

Due to the asynchronous nature of the handling of balancing re-connections, a re-connection might be initiated after the **JMSReplyTo** property has been translated to a fully qualified URI, but before the request message has been put to the request queue. Under these circumstances, the responding application might send its response to the original reply queue on the original queue manager, but the requesting application might now be waiting for a reply on the new queue manager.

Request_reply applications in a uniform cluster must therefore allow for the possibility of missed replies.

[Deploying flexible and scalable client applications](#)

[MQBNO - Balancing options](#)

Limitations and considerations for uniform clusters

Limitations and other points to consider when you are configuring uniform clusters.

Note: For general requirements when you are configuring uniform clusters, see also [“Creating a new uniform cluster”](#) on page 438.

Importance of uniformity between queue managers

By default, any application that declares itself as reconnectable might be rebalanced to an alternative queue manager in a uniform cluster at any time. This means that any resource, for example, queue, topic, or authority record that is required by such applications must be declared on all queue managers in the uniform cluster.

Consistency of queue manager configuration is not policed. It is up to your system administrator to configure members of the cluster so that they have a similar configuration.

However, you can aid consistency by using the Automatic configuration from an MQSC script at startup capability to share MQSC scripts that define objects for the cluster and therefore ensure that all have the same definitions. For more information, see [“Creating a new uniform cluster” on page 438](#).

This uniformity extends to full repository queue managers for the cluster. Although for traditional IBM MQ clusters it is often considered best practice to separate the full repositories onto stand-alone systems, in a uniform cluster, the model is that full repositories fully participate in the cluster and process application workloads alongside other nodes.

Overlapping uniform clusters and traditional IBM MQ clusters

A uniform cluster queue manager can participate in at most one uniform cluster, and it can also be a member of any number of standard IBM MQ clusters. It might be helpful to think of the uniform cluster as acting as a single queue manager in the wider cluster.

Note the following considerations:

- A uniform cluster queue manager acting as a full repository, must only be a full repository for the uniform cluster itself.
- Likewise, partial repository queue managers that are members of a uniform cluster, but might also belong to a wider traditional IBM MQ cluster, cannot be used as a repository outside the uniform cluster.

For more information, see [How to choose cluster queue managers to hold full repositories](#).

The reason is that queue managers that are full repositories for a combination of traditional IBM MQ clusters and uniform clusters, encourage a divergence of data held in the cluster cache across the members of the uniform cluster, and therefore move against the use of the uniform cluster feature as intended.

To replace a single full repository queue manager with a uniform cluster, separate the full repository from the application work that is in progress on it, and move only the application work into the uniform cluster.

When you are using automatic definitions for uniform clusters, cluster channels cannot be shared for use in other clusters, that is, you set the **CLUSTER** attribute to the automatic cluster, and the **CLUSNL** attribute must be empty.

Application balancing considerations

Application instances are not always evenly balanced, particularly in the following circumstances:

- When there are fewer application instances than queue managers in the cluster.
- During a short period after client applications connect to, or leave, the cluster.

To prevent client applications from being rebalanced too often, especially when applications connections are coming and going, limits are set on how often the uniform cluster asks client applications to be rebalanced. After a period of high connect or disconnect activity, it might take several minutes for the remaining application instances to be evenly balanced across the uniform cluster.

For more information, see [Application balancing troubleshooting](#).

Application affinities

Not all applications are suitable for automatic rebalancing across a uniform cluster. Only applications that specify **MQCNO_RECONNECT** are rebalanced. Applications that have an affinity to a particular queue manager must either specify the **MQCNO_NO_RECONNECT** option or **MQCNO_RECONNECT_Q_MGR**. The latter allows HA failover but not rebalancing.

Examples of applications that create an implicit affinity to a queue manager:

- Applications that create durable subscriptions.
- Applications that create permanent dynamic queues, for example, to receive reply messages.
- Applications that expect strict message ordering or require all messages in a sequence be processed by the same application instance or both.

These applications must specify **MQCNO_NO_RECONNECT** or **MQCNO_RECONNECT_Q_MGR** options rather than **MQCNO_RECONNECT**.

For more information, see [Reconnection options](#).

Message availability

While application balancing can rebalance connections around failed or temporarily unavailable queue managers, uniform clusters do not replicate message data across their members. For data availability, if a node fails, each member of the uniform cluster must also be configured to be highly available. Many data replication and high availability solutions are available, and can be combined with uniform clusters for maximum service and data availability, for example:

- Replicated storage that supports a container instance that is automatically restarted by container orchestration. For more information, see [Single resilient queue manager](#).
- RDQM queue managers. For more information, see [RDQM high availability](#).
- Multi instance queue managers. For more information, see [“Multi-instance queue managers” on page 540](#).
- Native HA. For more information, see [Native HA](#).
- IBM MQ Appliance HA. For more information, see [High availability](#).

Scalability and performance of uniform clusters

To enable the closer integration and sharing of application state between queue managers in a uniform cluster, a higher level of intercommunication is needed than in a traditional IBM MQ cluster. Therefore, scaling to large numbers of queue managers in a single uniform cluster is not recommended because the additional communication has detrimental effect on performance.

For both performance and management reasons, it is preferable to think of a uniform cluster as acting as a single traditional queue manager that provides messaging to a number of related applications, but is not a sole messaging service across an enterprise. In this pattern, small numbers, up to 10, queue managers are typically sufficient to support large numbers of client application connections. Application balancing makes it simple to start with small numbers, for example 3 queue managers, and scale up by adding more queue managers.



Attention: Enabling uniform cluster behavior in a cluster that does not have the recommended characteristics, in particular, use of clusters with large numbers of queue managers, is likely to have a severe performance impact.

Related concepts

[“Automatic application balancing” on page 425](#)

Automatic application balancing greatly enhances application distribution and availability by enabling an IBM MQ uniform cluster to closely manage application distribution across the cluster, rather than relying on randomization, or a manual pinning of applications to specific queue managers.

Creating a uniform cluster

You can simplify both the initial creation of a uniform cluster, and subsequently keeping the configuration between the uniform cluster members identical, by using the automatic configuration and automatic clustering support.

Before you begin

Before you create a uniform cluster, you should read [“Limitations and considerations for uniform clusters” on page 435](#).

About this task

You indicate that a particular IBM MQ cluster is to be treated as a uniform cluster by supplying in the `qm.ini` file a section for AutoCluster with at least **Type=Uniform** and **ClusterName=<uniform cluster name>**.

You can optionally configure the underlying IBM MQ cluster through the same `.ini` stanza by using *automatic cluster creation*. When using this automatic cluster support to set up the cluster, you provide a configuration file that describes the cluster and its full repositories.

If the queue manager being started is listed as one of the full repositories, then it is automatically made a full repository. Similarly, when the cluster receiver channel is defined, cluster sender channels to the full repository or repositories are automatically defined.

Procedure

To exploit additional function that requires a uniform cluster you must complete either of the following steps:

- [Convert an existing cluster to a uniform cluster](#), which meets the pattern described in [“About uniform clusters” on page 424](#).
- [Create a new uniform cluster](#) for this purpose.

Creating a new uniform cluster

How you create a new uniform cluster.

Procedure

1. Create a file that describes how you want the cluster itself to look in terms of full repositories.

As for any cluster, two full repositories act as central stores of information about the cluster.

Specifically, you need to describe the names and connection names for the two full repositories in this cluster.

Note: This is being done in advance of anything (including the queue managers) being created, and the following process below includes creating those queue managers.

For example, imagine you are setting up a uniform cluster called UNICLUS, with queue manager members QMA, QMB, QMC and QMD. In this example, QMA and QMB will be the full repositories, with QMC and QMD as partial repositories. A sample configuration file, `uniclus.ini`:

```
AutoCluster:
  Repository2Conname=QMA.dnsname(1414)
  Repository2Name=QMA
  Repository1Conname=QMB.dnsname(1414)
  Repository1Name=QMB
```

```
ClusterName=UNICLUS
Type=Uniform
```

The **RepositoryName** fields are used as the *conname* attribute for other cluster members to define cluster senders (CLUSDR) to them, and can be a connection list for a multi instance queue manager and can optionally include the port.

2. Create a sample configuration file, `uniclus.mqsc` which contains the MQSC definitions you want to be applied to all cluster members.

There is one mandatory line needed in this file, which is a definition of a cluster receiver channel (CLUSRCVR), with a CLUSTER attribute of the automatic cluster name (usually through the +AUTOCL+ insert) and a channel name which includes the +QMNAME+ insert.

This describes how other members of the uniform cluster connect to each queue manager and is used as a template of how to connect to the other queue managers as well. An example definition might be something like:

```
define channel('+AUTOCL+_QMNAME+') chltype(clusrcvr) trptype(tcp)
conname(+CONNAME+) cluster('+AUTOCL+') replace
```

When automatic clusters are configured, a definition of a cluster receiver channel can use some additional inserts in the CLUSTER, CONNAME, and CHANNEL fields to enable the definition to be identical on all queue managers in the uniform cluster. This includes:

+AUTOCL+

The automatic cluster name

+QMNAME+

The name of the queue manager being created

+CONNAME+

A variable defined during queue manager creation, using the **-iv** parameter, or in the Variables qm.ini stanza, for use in the connection name parameter string. The name of the variable can be any value.

Remember that channel names are limited to 20 characters, and so the value both with the inserts in, as well as when the inserts are replaced, needs to fit that limitation. An example file might look like the following:

```
*#####
* Compulsory section for all uniform cluster queue managers
*#####
define channel('+AUTOCL+_QMNAME+') chltype(clusrcvr) trptype(tcp) conname(+CONNAME+)
cluster('+AUTOCL+') replace
*
*#####
* Configuration for all queue managers
*#####
define QL(APPQ) maxdepth(99999999) replace
define QL(APPQ2) maxdepth(99999999) replace
define channel(CLIENTCHL) chltype(svrconn) trptype(tcp) replace
```

3. Make these two files available on each machine that will host a uniform cluster member.

For example, `/shared/uniclus.ini` and `/shared/uniclus.mqsc`.

4. On each of those machines, create the queue manager.

On the command line, supply:

- a. A request to start a listener, on the expected port
- b. A request for automatic INI configuration (**-ii**) pointing to the automatic cluster setup file (`uniclus.ini`)
- c. A request for automatic MQSC configuration (**-ic**) pointing to the MQSC configuration file which includes a CLUSRCVR definition for the uniform cluster.
- d. A CONNAME for this queue manager.

On the host for QMA:

```
crtmqm -p 1414 -ii /shared/uniclus.ini -ic /shared/uniclus.mqsc -iv  
CONNAME=QMA.dnsname(1414) QMA  
strmqm QMA
```

Each queue manager in the uniform cluster is created with an almost identical command line - all the differences between full and partial repository are handled automatically for a uniform cluster.

On the host for QMB:

```
crtmqm -p 1414 -ii /shared/uniclus.ini -ic /shared/uniclus.mqsc -iv  
CONNAME=QMB.dnsname(1414) QMB  
strmqm QMB
```

On the host for QMC:

```
crtmqm -p 1414 -ii /shared/uniclus.ini -ic /shared/uniclus.mqsc -iv  
CONNAME=QMC.dnsname(1414) QMC  
strmqm QMC
```

On the host for QMD:

```
crtmqm -p 1414 -ii /shared/uniclus.ini -ic /shared/uniclus.mqsc -iv  
CONNAME=QMD.dnsname(1414) QMD  
strmqm QMD
```

What happens automatically:

As the queue manager is started, the definitions from the `uniclus.ini` file are applied to the `qm.ini` file. For more information, see [“Automatic configuration of qm.ini at startup”](#) on page 108. This adds the **AutoCluster** definition into the `qm.ini` file.

If the queue manager is named in the **AutoCluster** stanza as one of the full repositories, then it is automatically converted to be a full repository, similar to issuing the MQSC command `ALTER QMGR REPOS(ClusterName)`, otherwise it is converted to a partial repository, similar to issuing the MQSC command `ALTER QMGR REPOS('')`.

When the define of the cluster receiver channel for the automatic cluster is processed, cluster sender channels are defined from this queue manager to all full repositories in the **AutoCluster** stanza (excluding the local queue manager if this is one of the full repositories). These sender channels inherit all common channel attributes from the local cluster receiver which has been defined.



Attention: Although the channels are created without further manual intervention, these are administrative channel objects which can be displayed and managed as for any other channel definition. You should not confuse these objects with 'auto-defined' cluster sender channels, created transiently and on-demand by the cluster to route message traffic.

What to do next

Verify the uniform cluster setup

When the **ClusterName** parameter is set correctly, and the queue manager is a member of the named cluster, message AMQ9883 is issued to confirm that the cluster is now identified as a uniform cluster.

You can then use uniform cluster capabilities, such as automatic application balancing. During queue manager startup, if this parameter has been set, but the name is not a valid IBM MQ cluster name, the name is ignored and error message AMQ9882 issued.

If the name is a valid cluster name, but no cluster channels for the identified cluster exist, warning message AMQ9881 is issued to the queue manager error log to allow your administrator to identify and correct this situation.

Verify the automated cluster setup

If you have used automatic cluster support to set up the uniform cluster, you can verify that the queue managers specified as full repositories are now correctly configured as such, using `runmqsc` commands:

```
QMA:
  1 : dis qmgr repos
AMQ8408I: Display Queue Manager details.
      QMNAME(QMA)                REPOS(UNICLUS)
```

Whereas partial repositories are not configured as repositories:

```
QMC:
  1 : dis qmgr repos
AMQ8408I: Display Queue Manager details.
      QMNAME(QMC)                REPOS( )
```

In addition, you should be able to see that cluster sender channels (CLUSSDR) have been configured from each queue manager to the other full repositories, using the channel name from the configuration MQSC file:

```
QMA:
  1 : dis chl(UNICLUS*) conname
AMQ8414I: Display Channel details.
      CHANNEL(UNICLUS_QMA)        CHLTYPE(CLUSRCVR)
      CONNAME(QMA.dnsname(1414))
AMQ8414I: Display Channel details.
      CHANNEL(UNICLUS_QMB)        CHLTYPE(CLUSSDR)
      CONNAME(QMB.dnsname(1414))

QMC:
  1 : dis chl(UNICLUS*) conname
AMQ8414I: Display Channel details.
      CHANNEL(UNICLUS_QMA)        CHLTYPE(CLUSSDR)
      CONNAME(QMA.dnsname(1414))
AMQ8414I: Display Channel details.
      CHANNEL(UNICLUS_QMB)        CHLTYPE(CLUSSDR)
      CONNAME(QMB.dnsname(1414))
AMQ8414I: Display Channel details.
      CHANNEL(UNICLUS_QMC)        CHLTYPE(CLUSRCVR)
      CONNAME(QMC.dnsname(1414))
```

Related concepts

[“About uniform clusters” on page 424](#)

The objective of a uniform cluster deployment is that applications can be designed for scale and availability, and can connect to any of the queue managers within the uniform cluster. This removes any dependency on a specific queue manager, resulting in better availability and workload balancing of messaging traffic.  Uniform clusters are not available on IBM MQ for z/OS; queue sharing groups provide many of the capabilities of a uniform cluster.

[“Limitations and considerations for uniform clusters” on page 435](#)

Limitations and other points to consider when you are configuring uniform clusters.

Converting an existing cluster to a uniform cluster

You can use this procedure to convert an existing cluster to a uniform cluster.

About this task

If you convert an existing cluster to a uniform cluster, you must ensure that any definition required to support applications balancing between queue managers exists on all cluster members.

Procedure

1. Enable IBM MQ publish subscribe, including remote (clustered) publish subscribe on all queue managers.

This is a prerequisite to uniform cluster functionality, so you must ensure that the PSMODE and PSCLUS attributes of the queue manager are both set to the default value of ENABLED.

2. Add an **AutoCluster** section to the qm. ini file to the name of the IBM MQ cluster, as used in your MQSC object definitions, such as cluster channels.

For example, if the name of the cluster is UNICLUS, add or modify the AutoCluster stanza in your `qm.ini` files as follows:

```
AutoCluster:
  ClusterName=UNICLUS
  Type=Uniform
```

3. Restart the queue managers to apply the new setting.
4. Consider using automatic configuration as a mechanism for ensuring all uniform cluster members have the same configuration applied from startup.

See [Automatic configuration from an MQSC script at startup](#) for more details.

Related concepts

[“About uniform clusters” on page 424](#)

The objective of a uniform cluster deployment is that applications can be designed for scale and availability, and can connect to any of the queue managers within the uniform cluster. This removes any dependency on a specific queue manager, resulting in better availability and workload balancing of messaging traffic.  Uniform clusters are not available on IBM MQ for z/OS; queue sharing groups provide many of the capabilities of a uniform cluster.

[“Limitations and considerations for uniform clusters” on page 435](#)

Limitations and other points to consider when you are configuring uniform clusters.

Using automatic cluster configuration

You configure IBM MQ to enable automatic configuration by changing the `qm.ini` configuration information.

Note: You can only use the AutoCluster stanza for uniform clusters.

Stanzas to configure

You can change the following stanzas:

AutoConfig

Defined in the `qm.ini` file. When the queue manager starts, it identifies which automatic configuration files to apply.

You should use this mechanism to distribute identical cluster configuration when uniform clusters are being used.

AutoCluster

Defined in the `qm.ini` file. Used when the queue manager starts to identify if the cluster is a member of an automatic cluster, and can identify the full repositories of the cluster.

Variables

Defined in the `qm.ini` file. Contains some queue manager variables.

Attributes for the AutoConfig stanza

The following two attributes are permitted in the AutoConfig stanza:

MQSCConfig=<Path>

The path is either a full file path, or path to a directory, where all files `*.mqsc` are applied to the queue manager, on every queue manager start.

For more information, see [Automatic configuration from an MQSC script at startup](#).

IniConfig=<Path>

The path is either a full file path, or path to a directory, where all files `*.ini` are applied to the `qm.ini` file, on every queue manager start.

For more information, see [“Automatic configuration of qm.ini at startup” on page 108](#).

These attributes are frequently used as part of setting up uniform clusters. For more information, see [“Creating a new uniform cluster”](#) on page 438.

Example stanza:

```
AutoConfig:
MQSCConfig=C:\MQ_Configuration\uniclus.mqsc
IniConfig=C:\MQ_Configuration\uniclus.ini
```

Attributes for the AutoCluster stanza

The following attributes are mandatory for the AutoCluster stanza:

Type=Uniform

Specifies the type of automatic cluster, and the only valid option is *Uniform*, which represents a uniform cluster.

ClusterName=<String>

Name of the cluster, that is the automatic cluster name.

The presence of the above attributes enables application balancing for uniform clusters. See [“Automatic application balancing”](#) on page 425 for further details.

In addition, simplified setup of a cluster can be performed if the cluster is described in this stanza. For more information, see [“Creating a new uniform cluster”](#) on page 438. When this is used, you can name two queue managers, and give their connection names for the full repositories for this automatic cluster.

The following attributes are optional for the AutoCluster stanza but you must provide them in pairs:

RepositoryName1=<String>

This is the queue manager name for the first full repository in the automatic cluster. This can be the name of this queue manager, or another one.

Repository1Conname=<Connection Name String>

This is the connection name (CONNNAME) value for how members of the automatic cluster should connect to this queue manager.

Additionally, you can identify a second full repository for the cluster:

Repository2Name=<String>

Repository2Conname=<Connection name string>

Example stanza:

```
AutoCluster:
Repository2Conname=myFR1.hostname(1414)
Repository2Name=QMFR1
Repository1Conname= myFR2.hostname(1414)
Repository1Name=QMFR2
ClusterName=UNICLUS
Type=Uniform
```

Attributes for the Variables stanza

An attribute=value pair is valid in the attribute field. These can be supplied using the **-iv** command line option on the **crtmqm** command when you create a queue manager.

You can use attributes listed in the Variables stanza during automatic cluster configuration of CONNAME and the channel name MQSC fields of a cluster receiver channel.

Suspending a queue manager from a uniform cluster

During normal operation of a uniform cluster, re-connectable client application instances might be automatically rebalanced at any time, to any queue manager in the cluster. If you want to prevent applications from connecting to a particular queue manager for a period of time, for example, during maintenance operations or problem determination, use the SUSPEND QMGR command.

Issue the command `SUSPEND QMGR CLUSTER(uniform cluster name)`

In addition to the usual effects of suspending from an IBM MQ cluster, in a uniform cluster, the SUSPEND command also prevents re-connectable applications from being rebalanced to this queue manager.

Any such existing connections to the queue manager are immediately rebalanced to other available queue managers in the cluster when the command is issued.

Notes:

- When queue managers are suspended from a cluster, `DIS APSTATUS` shows them as ACTIVE(NO), with the exception of the local queue manager, which always shows ACTIVE(YES) for its own status entry.
- If all queue managers in the uniform cluster are suspended, applications remain connected to one or more of the suspended queue managers.

To avoid new connections being added to the queue manager being maintained, you should stop the server connection channel, or channels, used by your client applications, for example by issuing the following **runmqsc** command:

```
STOP CHANNEL(svrconn channel name)
```

This might not be possible if, for example, these channels are also used to connect administrative applications required during the maintenance window. For this reason, the suspended queue manager periodically checks for attached re-connectable applications

If any re-connectable applications are present, they are rebalanced to other available queue managers in the cluster. Maintenance can now be performed on the suspended queue manager.

Note: Applications not considered movable are not affected by either the initial command or subsequent rescans, and remain connected to the suspended queue manager; see [MOVCOUNT](#) for more details.

To resume a suspended queue manager:

1. If required, start the server connection channel to resume accepting new application connections, by issuing the following command:

```
START CHANNEL(svrconn channel name)
```

2. Issue the following **runmqsc** command:

```
RESUME QMGR CLUSTER(uniform cluster name)
```

The queue manager resumes communication with the rest of the uniform cluster and, if required to restore balance, re-connectable client application instances are redirected to this 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.

Procedure

- See the following subtopics for more information on controlling queued publish/subscribe:
 - [“Setting queued publish/subscribe message attributes” on page 445](#)
 - [“Starting queued publish/subscribe” on page 446](#)
 - [“Stopping queued publish/subscribe” on page 446](#)
 - [“Adding a stream” on page 447](#)
 - [“Deleting a stream” on page 448](#)
 - [“Adding a subscription point” on page 448](#)
 - [“Combining topic spaces in publish/subscribe networks” on page 456](#)

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 28. Publish/subscribe configuration parameters	
Description	MQSC parameter name
Command message retry count	PSRTYCNT
Discard undeliverable command input message	PSNPMMSG
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 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

You start queued publish/subscribe by setting the PSMODE attribute of the queue manager.

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

You stop queued publish/subscribe by setting the PSMODE attribute of the queue manager.

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 IBM MQ 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 IBM 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 stream on the earlier version of IBM MQ.
2. Define a local topic with the same name as the stream on the earlier version of IBM MQ.
3. Add the name of the queue to the namelist, SYSTEM.QPUBSUB.QUEUE.NAMELIST
4. Repeat for all queue managers at the later version of IBM MQ that are in the publish/subscribe hierarchy.

Adding 'Sport'

In the example of sharing the stream 'Sport', the earlier version queue managers and the later version IBM MQ queue managers are working in the same publish/subscribe hierarchy. The earlier version queue managers share a stream called 'Sport'. The example shows how to create a queue and a topic on later version queue managers called 'Sport', with a topic string 'Sport' that is shared with the earlier version queue managers stream 'Sport'.

A later version queue manager publish application, publishing to topic 'Sport', with topic string 'Soccer/Results', creates the resultant topic string 'Sport/Soccer/Results'. On the later version queue managers, subscribers to topic 'Sport', with topic string 'Soccer/Results' receive the publication.

On earlier version queue managers, subscribers to stream 'Sport', with topic string 'Soccer/Results' receive the publication.

```
runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2026.  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.

You must configure each IBM MQ queue manager in the hierarchy manually.

Deleting a stream

You can delete a stream from an IBM MQ 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 IBM MQ 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 MQ, 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.

There is no need to add subscription points to use integrated publish/subscribe applications written for versions of IBM MQ.

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 333. 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**. 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 453.

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 458. You can also control which publications flow from one cluster to another. See [“Combining and isolating topic spaces in multiple clusters”](#) on page 460.

Related concepts

[Combining publication and subscription scopes](#)

From IBM WebSphere MQ 7.0 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](#)

[Troubleshooting distributed publish/subscribe problems](#)

[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](#)

From IBM WebSphere MQ 7.0 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 450](#).

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](#)

From IBM WebSphere MQ 7.0 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

From IBM WebSphere MQ 7.0 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](#)

From IBM WebSphere MQ 7.0 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

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 65 on page 459](#). 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.

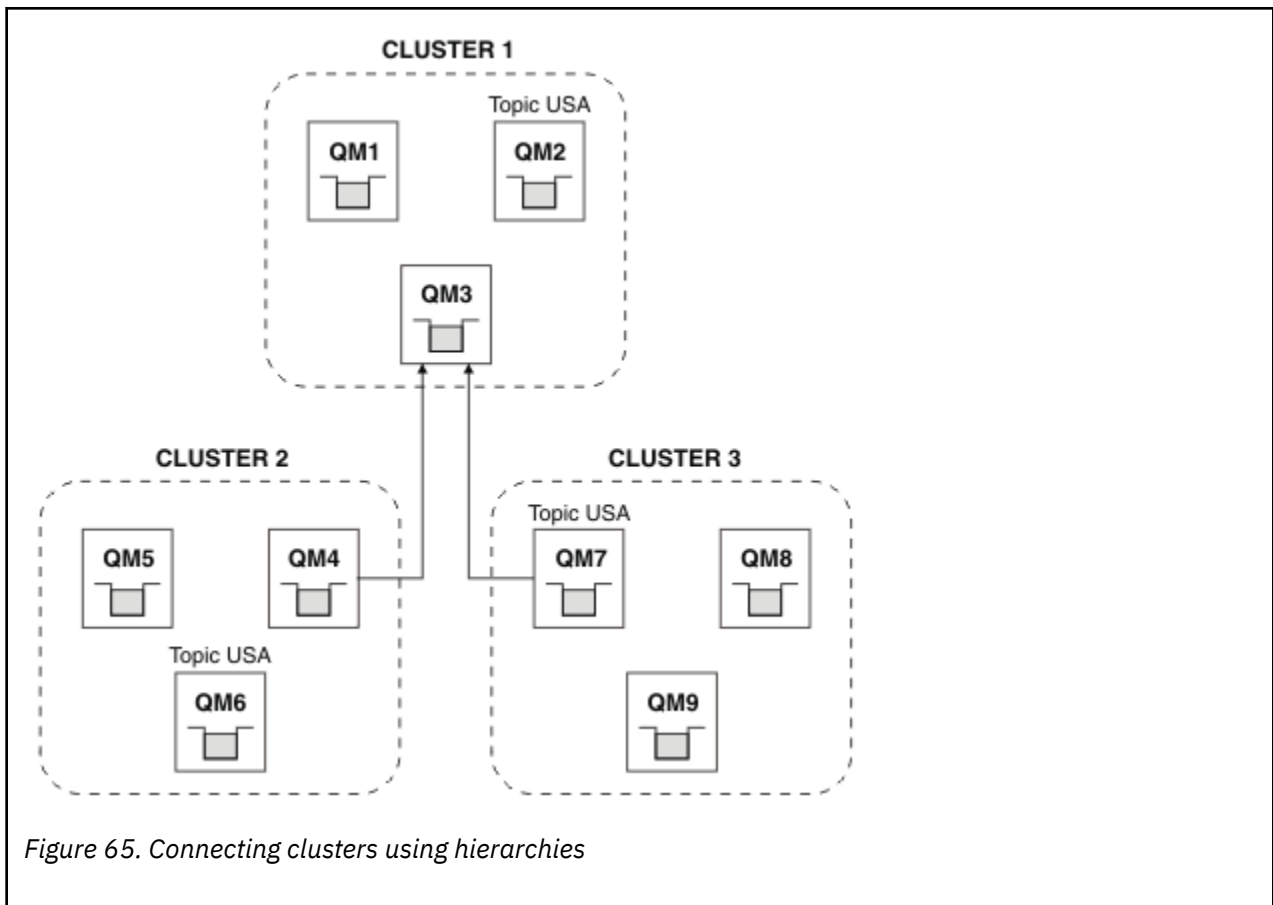


Figure 65. Connecting clusters using hierarchies

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 65 on page 459](#). 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 460](#).

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 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 458](#). 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 65 on page 459](#) in [“Combining the topic spaces of multiple clusters” on page 458](#) 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 66 on page 461](#). Use the bridging queue manager to filter which publications and subscriptions can flow from one cluster to another.

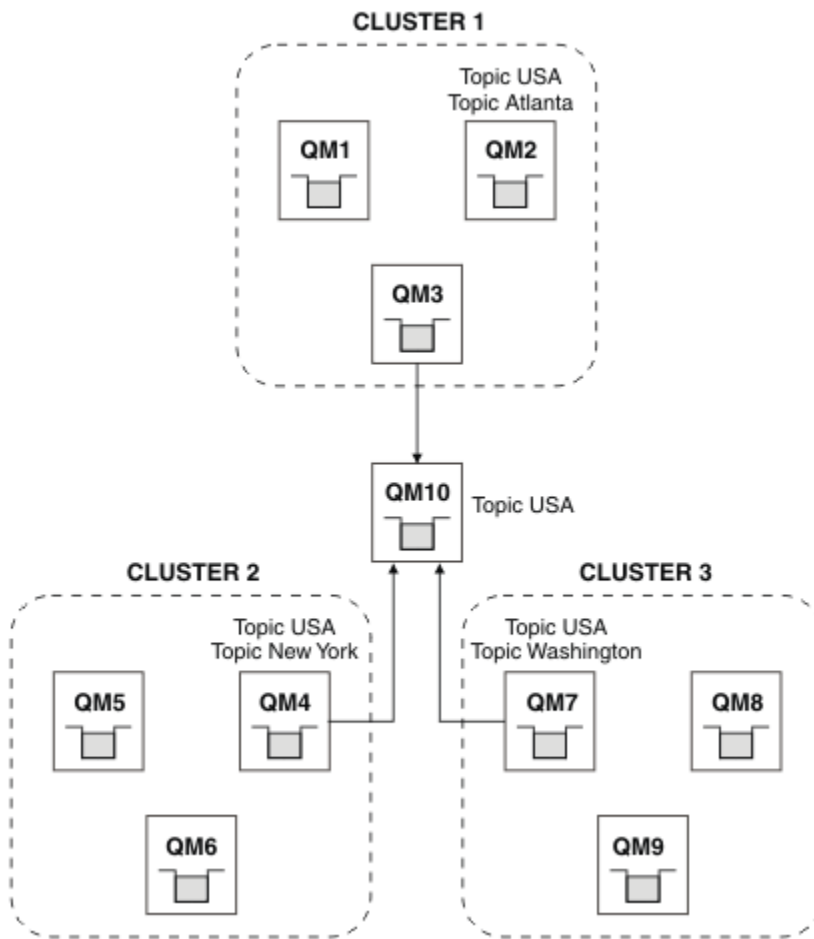


Figure 66. Bridged clusters

Use the bridge to isolate cluster topics that you do not want exposed across the bridge on the other clusters. In [Figure 66 on page 461](#), 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 66 on page 461](#), 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.

[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 67 on page 463](#). 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.

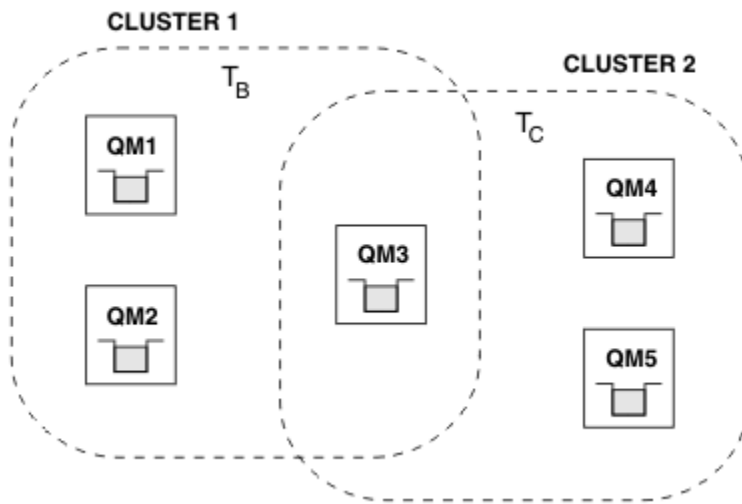


Figure 67. Overlapping clusters, non-overlapping topic spaces

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 67 on page 463](#) publishers and subscribers 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.

[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”](#) on page 446.
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 an IBM MQ cluster. See [“Adding a queue manager to a cluster”](#) on page 333.
 - 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”](#) on page 213.

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, 2026.  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, 2026.  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”](#) on page 449.

Related concepts

[Combining publication and subscription scopes](#)

From IBM WebSphere MQ 7.0 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.

[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

[Streams and topics](#)

[DISPLAY PUBSUB](#)

[Publish/subscribe messaging](#)

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, 2026.  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](#)

From IBM WebSphere MQ 7.0 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.

ALW

Configuring multiple installations

When using multiple installations on the same system, you must configure the installations and queue managers.

About this task

This information applies to AIX, Linux, and Windows.

Procedure

- Use the information in the following links to configure your installations:
 - [“Changing the primary installation” on page 474](#)
 - [“Associating a queue manager with an installation” on page 475](#)
 - [“Connecting applications in a multiple installation environment” on page 467](#)



Connecting applications in a multiple installation environment

On AIX, Linux, and Windows systems, if IBM MQ 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 MQ libraries” on page 468](#).

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 MQ 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.

Migration scenarios and connecting applications with multiple installations is considered in more detail in [Multi-installation queue manager coexistence on AIX, Linux, and Windows](#).

For more information about how to load IBM MQ libraries, see [“Loading IBM MQ libraries” on page 468](#).

Support and restrictions

If any of the following IBM MQ libraries are located and loaded, the product 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 471](#).

Related concepts

[“Restrictions for applications using multiple installations” on page 471](#)

There are restrictions when using CICS server libraries, fast path connections, message handles, and exits in a multiple installation environment.

[“Loading IBM MQ libraries” on page 468](#)

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 474](#)

You can use the **setmqinst** command to set or unset an installation as the primary installation.

[“Associating a queue manager with an installation” on page 475](#)

When you create a queue manager, it is automatically associated with the installation that issued the **crtmqm** command. On AIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

Loading IBM MQ 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 MQ libraries are located and loaded depends on your installation environment:

- On AIX and Linux systems, if a copy of an IBM MQ 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 an IBM MQ version installation to be the primary installation, or manually create the symbolic links.
- If IBM MQ is installed in a non-default location, you might need to change your existing applications so that the correct libraries are loaded.

How IBM MQ 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 470](#).

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 29. Benefits and drawbacks of the options for loading libraries

Platform	Option	Benefits	Drawbacks
 Linux  AIX AIX 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.
AIX and Linux systems	Set the <code>LD_LIBRARY_PATH</code> environment variable, using <code>setmqenv</code> , or <code>crtmqenv</code> , with the -k or -l option. ( On AIX, this environment variable is <code>LIBPATH</code>	• No changes to existing applications required. • Overrides embedded RPaths in an application. • Easy to change the variable if the location of IBM MQ changes.	• <code>setuid</code> and <code>setgid</code> applications, or applications built in other ways, might ignore <code>LD_LIBRARY_PATH</code> 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 <code>LD_LIBRARY_PATH</code>. •  Linux: The compiler used to build the application might disable the use of <code>LD_LIBRARY_PATH</code>. For more information, see Runtime linking considerations for Linux.
 Windows Windows systems	Set the <code>PATH</code> variable using <code>setmqenv</code> , or <code>crtmqenv</code> .	• 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.
 ALW AIX, Linux, and Windows systems	Set the primary installation to an IBM MQ, or later, installation. See “Changing the primary installation” on page 474 . 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.	•  Linux  AIX and Linux: Does not work if <code>/usr/lib</code> is not in the default search path.

Library loading considerations for Linux

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.

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.

Linux

AIX

On AIX 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). 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 MQ 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**
-  Linux: **readelf**

Related concepts

[“Restrictions for applications using multiple installations” on page 471](#)

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 467](#)

On AIX, Linux, and Windows systems, if IBM MQ 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 474](#)

You can use the **setmqinst** command to set or unset an installation as the primary installation.

[“Associating a queue manager with an installation” on page 475](#)

When you create a queue manager, it is automatically associated with the installation that issued the **crtmqm** command. On AIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

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 MQ 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 AIX, Linux, and Windows](#).

Fast path

On a server with multiple installations, applications using a fast path connection to IBM MQ 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 an IBM MQ installation, has no effect on whether an application can connect fast path.

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

[MQCONN - 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 472](#).

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 473](#).

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:

<i>Table 30. Configuring applications to use particular assemblies</i>		
	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

Table 30. 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 IBM MQ primary installation. (earlier version assemblies in GAC):	No changes necessary	In the application configuration file: Use the <i>bindingRedirect</i> 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 <i>codebase</i> tag to point to the location of the later version assemblies - Use the <i>bindingRedirect</i> tag to indicate the use of the later version assemblies	In the application configuration file: Use the <i>codebase</i> tag to point to the location of the later version assemblies
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 <i>codebase</i> tag to point to the location of the earlier version assemblies - Include the tag <i>publisherpolicy Apply=no</i>	In the application configuration file: - Use the <i>codebase</i> tag to point to the location of the earlier version assemblies - Use the <i>bindingRedirect</i> tag to indicate the use of the earlier version assemblies - Include the tag <i>publisherpolicy Apply=no</i>

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 467](#)

On AIX, Linux, and Windows systems, if IBM MQ 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 AIX, 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](#).

 During the installation process on Windows, you can specify that the installation is to be the primary installation.

  On AIX and Linux systems, you must issue a **setmqinst** command after installation to set the installation as the primary installation.

Procedure

- To set an installation as the primary installation, complete the following steps:
 - 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 an IBM MQ installation.

- If an existing IBM MQ installation is set as the primary installation, [unset it](#) before continuing with the next step.
- Make sure that you are logged on with the appropriate authority:
 -   As root on AIX and Linux.
 -  As 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
```

-  On Windows systems, restart the system.

- To unset an installation as the primary installation, complete the following steps:
 - 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 an IBM MQ installation.

b) Make sure that you are logged on with the appropriate authority:

-   As root on AIX and Linux.
 -  As 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 tasks

[Uninstalling, upgrading, and maintaining the primary installation](#)

[Choosing an installation name](#)

Related reference

[Features that can be used only with the primary installation on Windows](#)

[External library and control command links to primary installation on AIX and Linux](#)

[setmqinst](#)

ALW

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 AIX, Linux, and Windows, you can change the installation associated with a queue manager using the **setmqm** command.

About this task

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** has to be issued from the installation that is associated with the queue manager.
- **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.

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.

When you want to associate a queue manager with an installation, 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 [Maintaining and migrating](#).

Procedure

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.

What to do next

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 **dlmqm** command from that installation to delete the queue manager.

Related concepts

[“Finding installations of IBM MQ on a system” on page 476](#)

If you have multiple IBM MQ installations on a system, you can check which versions are installed and where they are.

[“IBM MQ configuration file, mqs.ini” on page 96](#)

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

[addmqinf](#)

[dspmq](#)

[dspmqinst](#)

[endmqm](#)

[setmqm](#)

[strmqm](#)

ALW

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 platform installation tools to query where IBM MQ has been installed. Then use the **dspmqr** command from an IBM MQ installation. The following commands are examples of commands you can use to query where IBM MQ has been installed:

- **AIX** On AIX systems, you can use the **lspp** command:

```
lspp -R ALL -l mqm.base.runtime
```

- **Linux** On Linux systems, you can use the **rpm** command:

```
rpm -qa --qf "%{NAME}-%{VERSION}-%{RELEASE}\t%{INSTPREFIXES}\n" | grep MQSeriesRuntime
```

- **Windows** 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
```

- **Linux** **AIX** On AIX 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 **dspmqr** command from an IBM MQ installation.

- **Windows** 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
```

- **Windows** On 64-bit Windows, issue the following command:

```
reg.exe query "HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\IBM\WebSphere MQ\Installation" /s
```

Related reference

[dspmqr](#)

[dspmqrinst](#)

[Multiple installations](#)

Configuring high availability and disaster recovery

If you want your applications to be highly available, all the components they depend on also need to be highly available, including any queue managers they connect to.

High availability (HA) solutions offer preemptive automatic failover. Such solutions provide redundancy to avoid single points of failure and are used to recover from localized resource failures. Disaster recovery (DR) solutions are reactive and typically require a configuration change to initiate a failover. Such solutions can be used to recover from site-wide failures. For maximum resiliency, it is possible to combine HA and DR solutions.

IBM MQ has a number of different mechanisms to enable queue managers to provide HA operation. The purpose of high availability operation is to quickly restore access to a given queue manager, and all the messages it contains. For this reason, it is sometimes referred to more specifically as “message availability”. High availability typically caters for situations where an individual component fails. Some of these HA mechanisms also provide for disaster recovery, where multiple components fail, up to and including loss of a whole region.

The primary high availability option is known as “native HA”. This covers the most commonly used platforms: container deployments and deployments directly on Linux. Other options include multi-

instance queue managers and replicated data queue managers (RDQM). Both Native HA and RDQM solutions can also provide disaster recovery.

IBM MQ clusters can also be used to help with availability, but these do not help with the recovery of a queue manager and the specific messages that it contains. They solve a slightly different problem, ensuring that there is always access to a queue manager that has the correct queue definitions, even if it does not contain all the actual messages. This enables applications to continue writing messages, even if they cannot read all the messages, and is sometimes referred to as “service availability”. IBM MQ clustering can be combined with highly available queue managers to provide the best of both message availability and service availability.

Native HA



Native HA provides high availability for IBM MQ in container deployments and also on Linux virtual machines or bare metal. It is the preferred HA solution because it has no dependencies on external components such as shared storage or operating system features.

Native HA uses log replication to keep three instances of a queue manager running on different nodes up to date. Only one instance is active at any time to process requests from applications. The active queue manager instance sends its log updates to the other two instances to keep them updated. If the active instance fails, one of the replica instances automatically takes over the active role.

Additionally, Native HA Cross-Region Replication (CRR) configuration enables you to switch the running of a queue manager to a different Native HA configuration in a different location. You can use a Native HA CRR configuration to provide a disaster recovery solution. If some disaster occurs at your live site, you can perform a failover and continue running the queue manager at the recovery site. Cross-Region Replication can also be used for planned failover to another site, for example, to perform maintenance.

 A Native HA Inter-Region Replication (IRR) configuration also enables you to switch the running of a queue manager to a different Native HA configuration in a different location and is also intended to provide a disaster recovery solution. The difference is that the IRR configuration is aimed at the situation where you have two data centers in the same region, and want to provide disaster recovery between them. See [“Comparison of Native HA CRR and Native HA IRR solutions” on page 498](#) for more information.

Note: Native HA In-Region Replication (IRR) is not currently supported in containers.

Multi-instance queue managers



Multi-instance queue managers use shared storage to enable two separate running instances to see the same queue data. Only one of the two instances is ever active and processing requests from applications. The two instances use the locking of a shared file to agree which is the active instance. Should the active instance fail, the other instance is able to take over the lock and become the active instance. Applications then automatically re-connect to the second instance and continue processing.

Multi-instance queue managers can be deployed on most platforms, but they must have low latency access to a suitable sophisticated shared storage capability.

Replicated data queue managers (RDQM)



Replicated data queue managers (RDQM) are only available for Red Hat Enterprise Linux because they rely on updates to Linux-specific kernel modules to perform data replication. Since IBM MQ 9.4.4, Native HA has been made available on Linux and is the preferred solution to RDQM in most circumstances because it does not have dependencies on the underlying operating system kernel. However, RDQM continues to be fully supported.

In RDQM, for high availability, three instances of the same queue manager are configured on three separate Linux servers. Only one of the three instances is the active instance. Data from the active queue manager is synchronously replicated to the other two instances, so one of these instances can take over in the event of some failure.

RDQM can also be used for disaster recovery. A queue manager runs on a primary node at one site, with a secondary instance of that queue manager located on a recovery node at a different site. Data is replicated between the primary instance and the secondary instance, and if the primary node is lost for some reason, the secondary instance can be made into the primary instance and started.

You can configure a replicated data queue manager (RDQM) that runs on a high availability group on one site, and can also fail over to another high availability group at another site if some disaster occurs that makes the first group unavailable. This is known as a RDQM HA/DR.

Note: You can use both the IBM MQ Console and IBM MQ Explorer with replicated data queue managers, but they do not display information that is specific to the RDQM features.

External HA solutions



It is also possible to use external high availability solutions with IBM MQ in order to create a highly available queue manager. Examples include the following solutions:

- Microsoft Cluster Server
- HA clusters on IBM i
- PowerHA® for AIX (formerly HACMP on AIX) and other UNIX and Linux clustering solutions

Note that, although many of these capabilities use the term “cluster”, they are very different from IBM MQ clusters described in [“IBM MQ clusters”](#) on page 479.

The external solutions provide mechanisms to bring back a specific queue manager on another machine, including its storage. As such they are functionally similar to the highly available queue manager mechanisms described earlier. However, using an external HA mechanism does have some key differences:

- Queue managers fail over more slowly than an HA mechanism internal to IBM MQ.
- External solutions are not integrated with IBM MQ tooling such as the IBM MQ Console or IBM MQ Explorer.
- External solutions require additional product purchase and deployment, and non-trivial additional skills.
- Use might require co-ordination with other teams.
- Require a deep knowledge of how IBM MQ works to ensure all relevant data and configuration are replicated consistently, and that IBM MQ is correctly monitored for activity/inactivity.

External HA clusters are sometimes chosen, however, where there is a need to coordinate multiple resources, such as an application server or database, in addition to IBM MQ.

IBM MQ clusters

IBM MQ clusters can be used to improve the resilience of queue managers, but what they offer is subtly different from what is provided by the high availability queue manager options described so far. IBM MQ clusters make it possible to have more than one queue manager offering the same queue definition at the same time. This means applications can connect to any of the active queue managers that have that queue, so this feature is sometimes described as providing “active/active” queue managers. If one of the queue managers has a failure, only some of the applications are affected, and these can rapidly re-connect to one of the active queue managers.

From a high availability point of view, IBM MQ clusters enable very high “service availability”, because an IBM MQ service is always available to the applications. They are always able to put new messages,

and they should always be able to receive at least some of the messages. However, when one of the queue managers in the cluster is down, all of its messages are unavailable. So IBM MQ clusters do not themselves improve “message availability”. For that you need to combine them with the HA queue manager options previously described.

z/OS



On z/OS, IBM MQ can directly take advantage of the high availability built into the platform. See [Shared queues and queue sharing groups](#).

IBM MQ Appliance



Another option for a high availability or disaster recovery solution is to deploy a pair of IBM MQ appliances. See [High Availability](#) and [Disaster Recovery](#) in the IBM MQ Appliance documentation.

IBM MQ as a Service

A final option to consider is the fully managed IBM MQ as a Service capability. In this model, the high availability of queue managers is operationally looked after by IBM. Queue managers are highly available by default, and can be spread across zones in a region for maximum resilience.

Native HA

Native HA is a high availability solution that is available on container deployments of IBM MQ and on Linux.

For information about Native HA on containers, see [Native HA in containers](#).

A Native HA configuration consists of three nodes, each with an instance of the queue manager. One instance is the active queue manager, processing messages and writing to its log. Whenever the log is written, the active queue manager sends the data to the other two instances, known as 'replicas'. Each replica writes to its own log, acknowledges the data, and then updates its own queue data from the replicated log. If the node running the active queue manager fails, one of the replica instances of the queue manager takes over the active role and has current data to operate with.

The following figure shows a typical deployment with three instances of a queue manager deployed on three nodes.

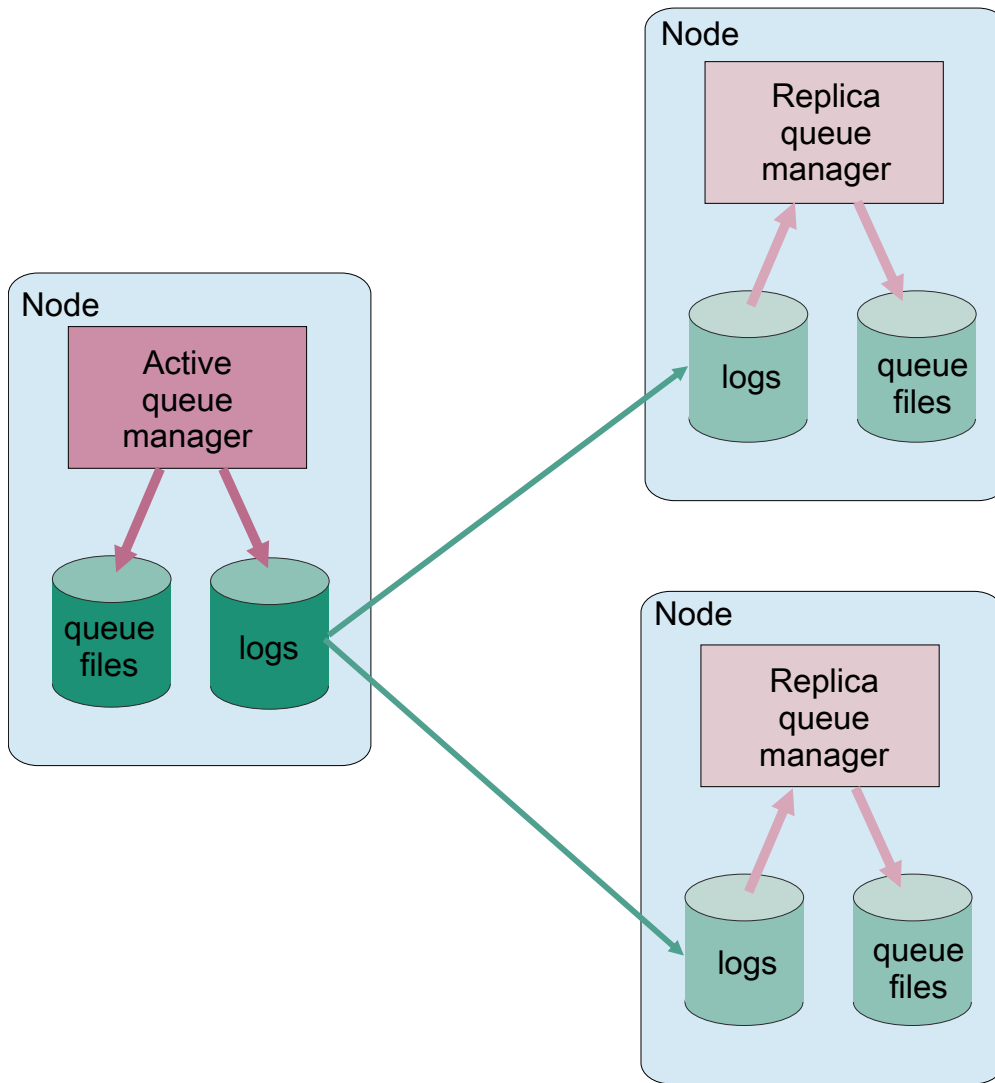


Figure 68. Example of Native HA configuration

Native HA on Linux

For a Linux implementation of Native HA, a node can be a single container (CRI-O, Docker, Podman, containerd), a Virtual Machine, or a bare-metal server that meets the current minimum system requirements for IBM MQ (see <https://www.ibm.com/support/pages/system-requirements-ibm-mq-94>).

All nodes must be deployed with the same processor architecture and located in the same geographical region. For the greatest levels of resiliency there should be no hardware resources shared between each node, and each node should be allocated the same amount of CPU, memory, and disk.

Each IBM MQ installation that hosts a Native HA instance must initially be using the same IBM MQ version and maintenance level and be installed in the same way, in the same locations. The minimum three packages required are:

- runtime (for example, `ibmmq-runtime_9.4.4.0_amd64.deb`)
- server (for example, `ibmmq-server_9.4.4.0_amd64.deb`)
- gskit (for example, `ibmmq-gskit_9.4.4.0_amd64.deb`)

The three nodes should have a consistent environment. For example, if a queue manager is configured to use a keystore in a specific location (such as `SSLKEYR(' /my/dir/keystore')`), you must ensure the same file location is mounted and accessible wherever the active queue manager instance runs.

You can use default locations for installation (`/opt/mqm`) and data (`/var/mqm`), but if you change the install or data location on one node, ensure you mirror the same change across all other nodes.

Liveness – monitor and automatic restart

V 10.0.0

Native HA uses the raft consensus algorithm to ensure data integrity and consistency. The instances collectively elect the best copy of the log available to lead the group through a majority decision. The elected leader of the Native HA group becomes the active instance, accepting IBM MQ application connections.

Log updates are sent from the active instance over the network to the other replica instances and once a majority (or 'quorum') number of copies of the log acknowledge the same log write, it can at that point be confirmed back to the application that requested it. The raft algorithm ensures through requiring majority decisions in elections that future leaders must have written this log record.

Should an active Native HA instance lose the ability to form quorum, for example, if it loses network connections to both the other instances, it is then unable to progress any work that requires log writes. Under these circumstances, the active instance abdicates (ends itself).

When running in a container environment using OpenShift or Kubernetes, there are liveness probes to monitor in the state of the queue manager instances, and restart them where necessary. The liveness probes are not available in a Linux configuration. When Native HA is configured on Linux, there are different ways to run a queue manager instance as a service to provide this functionality, one example is using `systemd` and an example configuration is provided using `systemd`, see [“Monitoring, restarting, and ending queue manager instances”](#) on page 484.

Readiness - locating the current active instance

V 10.0.0

When connecting clients and channels to a Native HA queue manager running outside of OpenShift or Kubernetes, there must be a way to connect clients or other queue managers to the active instance. You can use a comma-separated list of three possible addresses in order to locate the currently active instance.

Log configuration and media image scheduling

V 10.0.0

Native HA handles replication of the recovery log extents for a queue manager, this enables each replica to replay data from its locally maintained replica copy of the log and rebuild the configuration and state of the queue manager.

The size of a log extent is calculated by multiplying the log page size (4KB) by the number of log file pages (defined by `crtmqm -lf`). The number of log file pages can be in the range 64 to 65535 and defaults to 4096, which results in 16 MB log extents. The size of a log extent is fixed at the time a queue manager instance is created and cannot be changed. The number of log file pages must be identical on all instances that form a Native HA queue manager.

The number of primary (set by `crtmqm -lp`) and secondary (defined by `crtmqm -ls`) log extents determine the size of the active log which bounds long-running transactions. The size of the active log can be changed by altering the stanza values for primary and secondary extents in the `qm.ini` file. Any changes to the size of the active log must be implemented consistently across all three instances.

When sizing the log and log filesystem requirements for a Native HA queue manager, you can use the same approach as used to size a queue manager with linear logs (see [“Calculating the size of the log”](#) on page 733).

Provided that at least 100 messaging operations have been logged (for example, persistent puts, gets, commits, or rollbacks), a Native HA queue manager records a media image for each recoverable queue manager object every 60 minutes using the default IMGINTVL setting. A Native HA queue manager also sets IMGLOGLN to 25% of the available log filesystem space at the time the queue manager is created.

The IMGLOGLN setting works as a backstop and ensures that new media images are scheduled before the log gets too full when under high messaging workload. If creating a queue manager instance with a large log filesystem that might be shared by multiple queue managers, you should review the IMGLOGLN value and alter as necessary.

Instance naming

— V1000

Each instance must share the same queue manager name (together they form a single queue manager), but each instance must be given a unique name for identification purposes. When deploying Native HA in docker containers, virtual machines, or on bare-metal servers you can use the hostname as the instance name to identify the hosting location (but any three unique values would suffice, for example, athos, porthos, and aramis).

Instance names can be up to 48 characters in length. The names are case-sensitive and can include following valid characters:

- Uppercase (A-Z)
- Lowercase (a-z)
- Numerics 0-9
- Period (.)
- Forward slash (/)
- Dash (-)
- Underscore (_)
- Percent sign (%)

— V1000 Creating a Native HA queue manager

You need three nodes to create a Native HA queue manager with three instances on Linux. The nodes can be docker containers, virtual machines, or bare-metal servers, although you must use the same type of node for all three instances.

About this task

Each node must initially be able to communicate with the other nodes and start with the same IBM MQ maintenance level.

You create the three Native HA instances by using the **crtmqm** command on each node (although the three instances constitute a single queue manager).

In this example, the nodes have the host names alpha, beta, and gamma. For simplicity, these names are also used as the instance names inside the group. The group name is datacenter1.

The queue manager is called MYQMGR, and is created with ten primary and ten secondary extents of 32 MB. The -p option is specified to tell the queue manager to start a TCP/IP listener for channels and clients on port 1414, and start this when an instance is elected to start as an active.

After you create the three instances, you edit the qm.ini file for each instance to add information that lets them communicate with one another.

You can view the status of a Native HA queue manager by using the following methods:

- The **dspmq** command (see [dspmq \(display queue managers\)](#)).
- The IBM MQ Console (see [IBM MQ Console: Viewing queue manager status](#)).

- The IBM MQ Explorer (see [Native HA queue manager status attributes](#)).

Procedure

1. Create the first instance on the node named alpha:

```
crtmqm -lr alpha -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

The queue manager is created:

```
IBM MQ queue manager 'MYQMGR' created.
Directory '/var/mqm/qmgrs/MYQMGR' created.
The queue manager is associated with installation 'Installation1'.
Creating or replacing default objects for queue manager 'MYQMGR'.
Default objects statistics : 83 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

2. Create the second instance on the node named beta:

```
crtmqm -lr beta -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

3. Create the third instance on the node named gamma:

```
crtmqm -lr gamma -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

4. Add information to the qm.ini file for each instance:

```
NativeHAInstance:
  Name=alpha
  ReplicationAddress=alpha.example.com(9414)
NativeHAInstance:
  Name=beta
  ReplicationAddress=beta.example.com(9414)
NativeHAInstance:
  Name=gamma
  ReplicationAddress=gamma.example.com(9414)
```

5. You can now start the queue manager on each node:

```
strmqm MYQMGR
```

Results

After the instances have started, the first election takes place to choose the first leader and permanently set the QMId. This first election is special in that it requires all three instances to participate, whereas subsequent elections only require a majority to participate. The leader's log is replicated in plaintext over the network links to the two other replicas. When one of the two replicas is synchronized with the leader, there is quorum to begin writing to the log.

When the elected leader has quorum, it performs normal queue manager startup and becomes the active queue manager instance and opens for application connections. You can now connect applications and start using commands such as **runmqsc** to define and alter the settings of the queue manager.

Monitoring, restarting, and ending queue manager instances

You must implement a method to ensure that the queue manager instances in the Native HA configuration are still running, and restart them if required.

If a replica instance in a Native HA group unexpectedly fails, provided a majority of the instances are still available there is no loss of availability of a queue manager. It is, however, important to detect and restart the instance as quickly as possible. This topic provides an example of how you can implement a 'liveness' check using the **systemd** daemon. When you use this method, you must also use **systemd** to end the queue manager.

The following python script, `mqmonitor.py`, is provided in source form in `/opt/mqm/samp`. The script can start, monitor, stop, and restart a queue manager instance:

```
#
# <copyright
# notice="lm-source-program"
# pids="5724-H72"
# years="2026"
# crc="0" >
# Licensed Materials - Property of IBM
#
# 5724-H72
#
# (C) Copyright IBM Corp. 2026 All Rights Reserved.
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with
# IBM Corp.
# </copyright>
#
# A sample python script to start, monitor and stop a named queue manager.
#
import argparse
import subprocess
import time

def qmstart(qmgr, mqm_path):
    strmqm = subprocess.run([f"{mqm_path}/bin/strmqm", "-x", qmgr])
    # Allow multi-instance standby and Native HA replica instances also count as success
    if strmqm.returncode in [0, 30, 94]:
        print(f"Started {qmgr}")
    else:
        print(f"Failed to start {qmgr}, rc: {strmqm.returncode}")

def qmstop(qmgr, mqm_path):
    try:
        subprocess.run([f"{mqm_path}/bin/endmqm", "-tp", "29", "-w", qmgr], check=True)
        print(f"Stopped {qmgr}")
    except subprocess.CalledProcessError as e:
        print(f"Failed to stop {qmgr}: {e}")

def qmhealthy(qmgr, mqm_path):
    try:
        output = subprocess.check_output([f"{mqm_path}/bin/dspmq", "-n", "-m", qmgr],
            universal_newlines=True)
        active_states = ["STATUS(RUNNING)", "STATUS(STARTING)", "STATUS(RUNNING AS STANDBY)",
            "STATUS(NEGOTIATING)", "STATUS(REPLICA)", "STATUS(RECOVERY GROUP
LEADER)"]
        if any(x in output for x in active_states):
            return True
        else:
            return False
    except subprocess.CalledProcessError:
        return False

if __name__ == "__main__":
    MQ_INSTALLATION_PATH="/opt/mqm"
    parser = argparse.ArgumentParser(description="MQ Monitor Service")
    parser.add_argument("--start", action="store_true", help="Start monitoring queue manager")
    parser.add_argument("--stop", action="store_true", help="Stop queue manager and end
monitoring")
    parser.add_argument("qmgr", help="Name of queue manager to monitor")
    parser.add_argument("--mqm-path", default=MQ_INSTALLATION_PATH, help="MQ installation path
(default " + MQ_INSTALLATION_PATH + ")")
    parser.add_argument("--health-interval", type=int, default=10, help="Healthcheck interval
in seconds (default 10)")
    args = parser.parse_args()

    if args.start:
        while True:
            if not qmhealthy(args.qmgr, args.mqm_path):
                qmstart(args.qmgr, args.mqm_path)
            time.sleep(args.health_interval)
    elif args.stop:
        qmstop(args.qmgr, args.mqm_path)
    else:
        print("Usage: python3 mqmonitor.py --[start|stop] <qmgr> [--mqm-path <path>] [--health-
interval <seconds>]")
```

The `mqmonitor.py` python script can be combined with the following template system service definition `mqmonitor@.service`, which is also included in `/opt/mqm/samp`. Copying or symbolic linking the

definition into the `/etc/systemd/service` directory allows the service to be started by system. You must create the symbolic link as user `ROOT`.

```
#
# <copyright
# notice="lm-source-program"
# pids="5724-H72"
# years="2026"
# crc="0" >
# Licensed Materials - Property of IBM
#
# 5724-H72
#
# (C) Copyright IBM Corp. 2026 All Rights Reserved.
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with
# IBM Corp.
# </copyright>
#
# Copy or symlink this configuration file to /etc/systemd/service and use
#
# sudo systemctl enable mqmonitor@mqmname
#
# to configure a queue manager to start at system boot.
#

[Unit]
Description=MQ %I Queue Manager Monitor Service

[Service]
User=mqm
Group=mqm
Type=simple
StandardOutput=journal
StandardError=journal
SyslogIdentifier=mqmonitor
Environment=MQ_INSTALLATION_PATH=/opt/mqm
ExecStart=/usr/bin/python3 ${MQ_INSTALLATION_PATH}/samp/mqmonitor.py --mqm-path $
{MQ_INSTALLATION_PATH} --start %I --health-interval 10
ExecStop=/usr/bin/python3 ${MQ_INSTALLATION_PATH}/samp/mqmonitor.py --mqm-path $
{MQ_INSTALLATION_PATH} --stop %I
TimeoutStopSec=30
KillMode=mixed
Restart=always

[Install]
WantedBy=multi-user.target
```

If you are using the default install location `/opt/mqm` and do not want to modify the template or monitor script, you can symbolically link the sample service into place by using the following command:

```
ln -s /opt/mqm/samp/mqmonitor@.service /etc/systemd/system
```

You must create the symbolic link as user `ROOT`.

You then need to place the queue manager under the control of the `mqmonitor` service. For example, to place the `MYQMGR` queue manager under the control of the service, issue the following commands:

```
sudo systemctl enable mqmonitor@MYQMGR
sudo systemctl start mqmonitor@MYQMGR
```

When running under `systemd` control, the python script wakes every ten seconds and issues a **dspmq** command to check that the queue manager instance is in a useful state. If the state is unacceptable, the python script attempts to restart the queue manager.

Ending a Native HA queue manager

A queue manager under **systemd** control, is checked every ten seconds by the python script and restarted if it is not in an unacceptable state. Therefore, issuing `endmqm` for the queue manager results in

the python script restarting the queue manager. Instead, if you want to stop the queue manager, use the `systemctl stop` command:

```
sudo systemctl stop mqmonitor@MYQMGR
```

Use `systemctl` with the `start` or `restart` options to start the queue manager.

Adding TLS encryption

You can add TLS encryption to the connections between the queue manager instances in a Native HA configuration.

About this task

The replication traffic between instances carries the content of the queue manager log, which might contain sensitive business data from application messages. For this reason, it is recommended to set up TLS encryption in your Native HA configuration.

These instructions show you to set up TLS in the example configuration that was created in [“Creating a Native HA queue manager”](#) on page 483.

Procedure

1. On the host named alpha, run the following two commands to first create a keystore and then a self-signed certificate:

```
runmqakm -keydb -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw passw0rd -stash
runmqakm -cert -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw passw0rd -label
selfsigned -dn CN=MYQMGR -size 2048
```

These commands create the following four files:

- keystore.crl
 - keystore.kdb
 - keystore.rdb
 - keystore.sth
2. Copy the four files to the same location (`/var/mqm/qmgrs/MYQMGR/ssl`) on the beta and gamma hosts, for example, by using **sftp**.
 3. On each host (alpha, beta, and gamma), ensure that the files have mqm group ownership and have read access for the mqm group:

```
chown :mqm /var/mqm/qmgrs/MYQMGR/ssl/keystore.*
chmod g+r /var/mqm/qmgrs/MYQMGR/ssl/keystore.*
```

4. Instruct the Native HA configuration to use the self-signed certificate in the keystore with the strongest TLS 1.2 cipher. On each host in turn (alpha, beta, and gamma), edit the `qm.ini` file and add the following text to the `NativeHALocalInstance` stanza:

```
NativeHALocalInstance:
  CipherSpec=ANY_TLS12
  CertificateLabel=selfsigned
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore
```

5. After the configuration change is made on each host, restart each instance, for example, by restarting the mqmonitor service for the MYQMGR queue manager.

```
sudo systemctl restart mqmonitor@MYQMGR
```

When a majority of the group have restarted and are running with the TLS configuration a new active instance is elected.

6. On any of the hosts (alpha, beta, or gamma), confirm that each instance is connected by using the **dspmq** command.

```
dspmq -m MYQMGR -o nativeha -x
```

When all three instances are connected, the status QUORUM(3/3) is shown. The AMQ3212 and AMQ3214 messages are also written in the queue manager error logs, indicating that TLS is being used and stating which CipherSpec was negotiated.

Example: Deploying a simple Native HA configuration on Linux

This example installs a queue manager using a simple Native HA configuration across three Linux hosts.

Before you begin

To complete this example, you must first have completed the following prerequisites:

- Follow the steps in [Checking requirements on Linux](#).
- Follow the steps in [Planning to install IBM MQ on Linux](#)
- For each of the three Linux hosts, follow the steps in [Preparing the system on Linux](#).
- Download the 10.0.0 or above IBM MQ base install images for your Linux distribution (see [IBM MQ downloads](#)).

About this task

The Native HA configuration defines a group of instances, which together form a highly available queue manager. Each instance is installed on a different host, so if one host fails the others can take over.

This example defines a single Native HA group consisting of three IBM MQ instances installed on their own Linux hosts. The host names of the instances are alpha, beta, and gamma. Throughout the example, these names should be replaced with the host names of your systems.

This example uses the default installation location (/opt/mqm) and data directory (/var/mqm). If you are using non-default locations, be sure to replace these paths accordingly throughout the example.

See “Native HA” on page 480 for more details on Native HA configurations. Native HA also supports Cross Region Replication (CRR), where an additional Native HA group acts as a recovery group, see “Native HA CRR” on page 499 for more information.

This example uses a single certificate to secure the queue manager instance traffic.

This example walks through defining a Native HA group on Linux hosts to provide high availability, setting up monitoring and restarts, and forcing a failover from the active instance to a replica.

Procedure

1. Install IBM MQ on all three Linux hosts.

Follow the instructions in [Installing and uninstalling IBM MQ on Linux](#) to accept the IBM MQ license and install the IBM MQ server on all three of your hosts.

For this example, it is easiest to use the default locations for installation (/opt/mqm) and data (/var/mqm). If you change the install or data location on one system, ensure that you mirror the same change across all other hosts.

2. Set up environments ready to install the Native HA instances:

- a) On each Linux host, log in as the mqm user and set up your environment by using the following command:

```
. /opt/mqm/bin/setmqenv -s
```

- b) Check that the environment is set up correctly by entering the following command:

```
dspmqrver
```

If the command completes successfully, and the expected version number and installation name are returned, the environment is set up correctly.

3. Create the Native HA queue manager instances. On hosts alpha, beta, and gamma, create a queue manager with log replication. (You might want to use the AutoConfig stanza to ensure objects are always defined in a uniform way, the **-ic** option can be added to **crtmqm** to supply a mqsc file for the queue manager.)

- a) Create the first instance on the node named alpha:

```
crtmqm -lr alpha -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

The queue manager is created:

```
IBM MQ queue manager 'MYQMGR' created.  
Directory '/var/mqm/qmgrs/MYQMGR' created.  
The queue manager is associated with installation 'Installation1'.  
Creating or replacing default objects for queue manager 'MYQMGR'.  
Default objects statistics : 83 created. 0 replaced. 0 failed.  
Completing setup.  
Setup completed.
```

- b) Create the second instance on the node named beta:

```
crtmqm -lr beta -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

- c) Create the third instance on the node named gamma:

```
crtmqm -lr gamma -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

4. Create a keystore for securing the instance and group traffic:

- a) To create this keystore, log on to the alpha host and run the following commands:

```
runmqakm -keydb -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -stash  
runmqakm -cert -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label  
nha-qm-replication -dn CN=MYQMGR-REPLICATION -size 2048
```

Where PASSWORD is a password of your own choosing.

- b) Copy the four keystore files to the same location (`/var/mqm/qmgrs/MYQMGR/ssl/`) on the beta and gamma hosts, for example, by using sftp.
- c) On each host (alpha, beta, and gamma), ensure that the files have mqm group ownership and have read access for the mqm group:

```
chown :mqm /var/mqm/qmgrs/MYQMGR/ssl/keystore.*  
chmod g+r /var/mqm/qmgrs/MYQMGR/ssl/keystore.*
```

Important: The examples described here are not suitable for a production environment, and are solely intended as examples to get going quickly. Certificate management is a complex subject for advanced users. For production, you must consider things like rotation, revocation, key length, disaster recovery and much more.

5. Create the Native HA group. The group is created by editing the `/var/mqm/qmgrs/MYQMGR/qm.ini` files for each queue manager instance.

- a) Update the NativeHALocalInstance stanza for each of the alpha, beta, and gamma instances to define the group and the certificate to use to secure the instance traffic:

```
NativeHALocalInstance:  
  Name=<instance-name>  
  CipherSpec=ANY_TLS12  
  CertificateLabel=nha-qm-replication  
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore
```

Where:

- `<instance-name>` is the name of the instance that you are configuring (that is, alpha, beta, or gamma).
- `CertificateLabel` specifies the certificate for traffic between instances.

b) For each of the alpha, beta, and gamma instances add the local group members:

```
NativeHAInstance:
  Name=alpha
  ReplicationAddress=alpha.example.com(9414)
NativeHAInstance:
  Name=beta
  ReplicationAddress=beta.example.com(9414)
NativeHAInstance:
  Name=gamma
  ReplicationAddress=gamma.example.com(9414)
```

(Replace host names and replication addresses with the names and replication addresses of your own Linux nodes.)

c) Start the instances in the group. You can now start the queue manager instances on alpha, beta, and gamma hosts:

strmqm MYQMGR

d) Verify the group has deployed correctly. Issue the **dspmq -o nativeha -x** command on the alpha, beta, and gamma hosts, to display the group status. On the active instance, the status is:

```
QMNAME(MYQMGR)                                ROLE(Active)
INSTANCE(<instance-name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME()
GRPROLE(Live)
  INSTANCE(alpha) ROLE(Active) REPLADDR(<alpha-fqdn>) CONNACTV(yes) INSYNC(yes)
  BACKLOG(0) CONNINST(yes) ACKLSN(<n:n:n:n>) HASTATUS(Normal) SYNCTIME(<timestamp>)
  ALTDATE(<date>) ALTTIME(<time>)
  INSTANCE(beta) ROLE(Replica) REPLADDR(<beta-fqdn>) CONNACTV(yes) INSYNC(yes) BACKLOG(0)
  CONNINST(yes) ACKLSN(<n:n:n:n>) HASTATUS(Normal) SYNCTIME(<timestamp>) ALTDATE(<date>)
  ALTTIME(<time>)
  INSTANCE(gamma) ROLE(Replica) REPLADDR(<gamma-fqdn>) CONNACTV(yes) INSYNC(yes)
  BACKLOG(0) CONNINST(yes) ACKLSN(<n:n:n:n>) HASTATUS(Normal) SYNCTIME(<timestamp>)
  ALTDATE(<date>) ALTTIME(<time>)
```

The key parts of the status are:

- `ROLE(Active)`
- `QUORUM(3/3)` - it has full quorum, note this can take a few seconds

Note which host in the group has the Active role. The other hosts will have a similar status, but the `ROLE` is `Replica`.

6. Add monitoring and restarting of the queue manager.

If an instance unexpectedly fails, it is important to detect the failure and restart the instance as quickly as possible. In this example, a sample demonstrates the implementation of a 'liveness' check. The sample uses a python script and systemd daemon. When you use this method, you must use `systemd` to start and end the queue manager instead of the IBM MQ commands.

Follow the instructions in [“Monitoring, restarting, and ending queue manager instances” on page 484](#) to add queue manager monitoring and restarting to the alpha, beta, and gamma hosts.

7. Define a queue. On the active instance, start MQSC by entering the following command:

```
runmqsc MYQMGR
```

A message indicates when MQSC starts. MQSC has no command prompt. Define a local persistent queue:

```
DEFINE QLOCAL('QUEUE1') DEFPSIST(YES)
```

A message indicates when the queue is created. Stop MQSC by entering the following command:

```
end
```

Messages are shown, followed by the command prompt. The QUEUE1 queue is replicated to the other instances in the group.

8. Add persistent messages to the queue. The DEFPSIST (YES) attribute on the QUEUE1 queue causes messages put to the queue with the amqsputc sample to be persistent. They are replicated to the replica instances.

Change directory into /opt/mqm/samp/bin, which contains the sample programs. Put messages on the queue by entering the following commands:

```
./amqsput QUEUE1 MYQMGR
```

If connection to the queue manager is successful, the following response is output:

```
target queue is QUEUE1
```

Put several messages to the queue, by entering some text then pressing Enter each time. Enter a blank line to end the message input. The following message is shown:

```
Sample AMQSPUT0 end
```

Your messages are now on the queue and the command prompt is shown.

9. Restart the active to mimic a host failure. Stop the active instance, wait a second and then start the active using the following command:

```
sudo systemctl stop mqmonitor@MYQMGR  
sudo systemctl start mqmonitor@MYQMGR
```

10. Locate the new active instance. Issue the **dspmq -o nativeha -x** command on alpha, beta, and gamma hosts, to display the group status and identify which host is now the active instance.
11. Retrieve messages sent before the active instance was ended. On the current active instance, change directory into /opt/mqm/samp/bin, which contains the sample programs. Get the messages from the queue by entering the following command:

```
./amqsget QUEUE1 MYQMGR
```

The messages you added in step 8 are available, and are output. After a few seconds, the command exits.

What to do next

You could also try adding a recovery group to this group to provide cross region replication using the instructions at [“Example: Deploying a simple Native HA CRR configuration on Linux” on page 511.](#)

Example: Adding a recovery group to an existing Native HA configuration on Linux

This example builds on the previous example that created a Native HA group. This example adds a recovery group, extending the configuration to be a Native HA Cross-Region Replication (CRR) configuration.

Before you begin

To complete this example, you must first have completed the exercise [“Example: Deploying a simple Native HA configuration on Linux” on page 488](#), and also met the following prerequisites:

- Follow the steps in [Checking requirements on Linux](#).
- Follow the steps in [Planning to install IBM MQ on Linux](#)
- For each of the three Linux hosts, follow the steps in [Preparing the system on Linux](#).
- Download the 10.0.0 or above IBM MQ base install images for your Linux distribution (see [IBM MQ downloads](#)).

About this task

The Native HA CRR configuration defines two Native HA groups, which together form a highly available queue manager. Usually these groups are created in two different regions to provide the cross region replication. However, there are no constraints on where groups can be located and for this example, it is easier to create both groups on Linux hosts in a single location.

This example defines a London and a Rome group, each consisting of three IBM MQ instances installed on their own Linux host. The instances alpha, beta, gamma were created as a Native HA group in [“Example: Deploying a simple Native HA configuration on Linux”](#) on page 488. They are defined as the London group in this example. The instances delta, epsilon, and zeta will be used to create the Rome group. Throughout the example these host names should be replaced with the host names of your machines.

This example uses the default installation location (/opt/mqm) and data directory (/var/mqm). If you are using non-default locations, be sure to replace these paths accordingly throughout the example.

See [“Native HA CRR”](#) on page 499 for a more detailed overview.

For simplicity, this example uses a single certificate to secure all the queue manager Native HA instance and Native HA CRR group to group traffic.

To understand how certificates are used in Native HA CRR and how to use different certificates to secure group and instance traffic, see [“Example: using multiple TLS certificates with Native HA CRR on Linux”](#) on page 517.

This example walks through:

- Creating an additional Native HA group on three Linux hosts
- Changing the previously created group to a Live group
- Adding the new group as a Recovery group to provide cross-region replication
- Using a sample to add monitoring and restarts
- Demonstrate a switchover

Procedure

1. If you haven't already done so, follow the instructions in [“Example: Deploying a simple Native HA configuration on Linux”](#) on page 488 to configure the alpha, beta, and gamma hosts as a Native HA queue manager.
2. Install IBM MQ on three more Linux hosts.

Follow the instructions in [Installing and uninstalling IBM MQ on Linux](#) to accept the IBM MQ license and install the IBM MQ server on the delta, epsilon, and zeta hosts.

For this example, it is easiest to use the default locations for installation (/opt/mqm) and data (/var/mqm). If you change the install or data location on one system, ensure that you mirror the same change across all other hosts.

3. Set up environments ready to install the Native HA instances.
 - a) On the delta, epsilon, and zeta hosts,, log in as the mqm user and set up your environment by using the following command:

```
. /opt/mqm/bin/setmqenv -s
```

- b) Check that the environment is set up correctly by entering the following command:

```
dspmqrver
```

If the command completes successfully, and the expected version number and installation name are returned, the environment is set up correctly.

4. Create the Rome queue manager instances. As the mqm user, on hosts delta, epsilon, and zeta create a queue manager with log replication. (If you used the AutoConfig stanza for the London instances, use the **crrmqm -ic** option for these instances as well.)

a) Create the first Rome instance on the node named delta:

```
crtmqm -lr delta -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

b) Create the second Rome instance on the node named epsilon:

```
crtmqm -lr epsilon -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

c) Create the third Rome instance on the node named zeta:

```
crtmqm -lr zeta -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

5. Adding a keystore for securing the Rome instance and group traffic. Copy the four keystore files created in Step 4 of [“Example: Deploying a simple Native HA configuration on Linux”](#) on page 488 to the `/var/mqm/qmgrs/MYQMGR/ssl/` folder on the delta, epsilon, and zeta hosts, for example, by using `sftp`.
6. On each host (delta, epsilon, and zeta), ensure that the files have mqm group ownership and have read access for the mqm group:

```
chown :mqm /var/mqm/qmgrs/MYQMGR/ssl/keystore.*
chmod g+r /var/mqm/qmgrs/MYQMGR/ssl/keystore.*
```

Important: The examples described here are not suitable for a production environment, and are solely intended as examples to get going quickly. Certificate management is a complex subject for advanced users. For production, you must consider things like rotation, revocation, key length, disaster recovery and much more.

7. Change the original Native HA group created on the alpha, beta, and gamma instances to be the London group of the Native HA CRR configuration. The London group will initially have the Live role. The group is created by editing the `/var/mqm/qmgrs/MYQMGR/qm.ini` files for each queue manager instance.
 - a) Update the `NativeHALocalInstance` stanza for each of the alpha, beta, and gamma instances to define the group. Keep certificate information that was added when the group was initially created to secure the instance traffic:

```
NativeHALocalInstance:
  Name=<instance-name>
  GroupName=london
  GroupRole=Live
  GroupLocalAddress=(9415)
  CipherSpec=ANY_TLS12
  CertificateLabel=nha-qm-replication
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore
```

Where:

- `<instance-name>` is the name of the instance that you are configuring (that is, alpha, beta, or gamma).
- `CertificateLabel` specifies the certificate for traffic between instances. In this example, the same certificate is used for the traffic between instances and groups. (The `GroupCertificateLabel` specifies the certificate from group to group traffic. When it is not specified, the groups use the same certificate as the instance traffic.)

b) For each of the alpha, beta, and gamma instances add the recovery group members:

```
NativeHARecoveryGroup:
  GroupName=rome

  ReplicationAddress=delta.example.com(9415),epsilon.example.com(9415),zeta.example.com(9415)
  Enabled=Yes
```

(Replace host names and replication addresses with the names and replication addresses of your own Linux nodes.)

- c) Restart the queue manager instances on alpha, beta, and gamma hosts so the new configuration is deployed:

```
sudo systemctl restart mqmonitor@MYQMGR
```

- d) Verify that the London group has deployed correctly. Issue the **dspmqr -o nativeha -g** command on the alpha, beta, and gamma hosts, to display the group status. On the active instance, the status is:

```
QMNAME(MYQMGR)                                ROLE(Active) INSTANCE(<instance-  
name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME(london) GRPROLE(Live)  
GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) GRSTATUS(Normal)  
RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>) INITTIME(<timestamp>)  
LIVETIME(<timestamp>) ALTDATETIME(<date>) ALTTIME(<time>)  
GRPNAME(rome) GRPROLE(Unknown) GRPADDR(Unknown) GRPVER(?.?.?.?) CONNGRP(no)  
GRSTATUS(Unknown) RCOVLSN() RCOVTIME() ALTDATETIME() ALTTIME()
```

The key parts of the status are:

- **ROLE(Active)**
- **GRPNAME(london) GRPROLE(Live)** - London is the live group
- **QUORUM(3/3)** - it has full quorum
- **GRPNAME(rome)** - the recovery group will be in Rome. (The GRPADDR is unknown as the Rome group does not exist yet.)

Take a note of which instance in the London group is currently the active one. The other hosts have similar status, but the **ROLE** is **Replica**.

- e) On the active instance, check `/var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG` for the following messages to confirm the connections between the instances are secured:

```
AMQ3305I: The Native HA secure connection between the local instance and '<other-  
instance-1>' is using certificate DN 'CN=MYQMGR-REPLICATION'.  
AMQ3305I: The Native HA secure connection between the local instance and '<other-  
instance-2>' is using certificate DN 'CN=MYQMGR-REPLICATION'.
```

8. Create the Rome group on the delta, epsilon, and zeta hosts.

- a) Edit the `/var/mqm/qmgrs/MYQMGR/qm.ini` file for each host. Add the following to the **NativeHALocalInstance** stanza:

```
NativeHALocalInstance:  
  Name=<instance-name>  
  GroupName=rome  
  GroupRole=Recovery  
  GroupLocalAddress=(9415)  
  CipherSpec=ANY_TLS12  
  CertificateLabel=nha-qm-replication  
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore
```

- **<instance-name>** is the name of the instance that you are configuring (that is, delta, epsilon, or zeta).

- b) For each of the delta, epsilon, and zeta hosts add the local and recovery group members:

```
NativeHAInstance:  
  Name=delta  
  ReplicationAddress=delta.example.com(9414)  
NativeHAInstance:  
  Name=epsilon  
  ReplicationAddress=epsilon.example.com(9414)  
NativeHAInstance:  
  Name=zeta  
  ReplicationAddress=zeta.example.com(9414)  
NativeHARecoverGroup:  
  GroupName=london  
  
ReplicationAddress=alpha.example.com(9415),beta.example.com(9415),gamma.example.com(9415)  
Enabled=Yes
```

Where:

- The NativeHAInstance stanzas define all the instances in the Rome Native HA group.
- The NativeHARecoveryGroup stanza provides details about the current live group and its members (these details are not used until Rome becomes the live group and London becomes the recovery group).

(Replace host names and replications addresses with the names and replication addresses of your own Linux nodes.)

- c) Start the instances in the Rome group. You can now start the queue manager instances on delta, epsilon, and zeta hosts:

```
stimqm MYQMGR
```

- d) Verify the Rome group has deployed correctly. Issue the **dspmq -o nativeha -g** command on the delta, epsilon, and zeta instances, to get their group status. On the leader, the status is:

```
QMNAME(MYQMGR)                                ROLE(Leader) INSTANCE(<instance-  
name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME(rome) GRPROLE(Recovery)  
GRPNAME(rome) GRPROLE(Recovery) GRPADDR(<leader-fqdn>) GRPVER(10.0.0.0)  
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) BACKLOG(0) INSYNC(yes)  
SYNCTIME(<timestamp>) ALTDATE(<date>) ALTTIME(<time>)  
GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) CONNGRP(yes)  
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>)  
INITTIME(<timestamp>) LIVETIME(<timestamp>) ALTDATE(<date>) ALTTIME(<time>)
```

It might take a few seconds to reach full quorum. The key parts of this status are:

- ROLE(Leader)
- GRPNAME(rome) GRPROLE(Recovery) - Rome is the recovery group
- QUORUM(3/3) - it has full quorum, note this can take a few seconds
- GRPNAME(london) - London is the live group

- e) Issue the **dspmq -o nativeha -g** command on the active node of the London group, the status is now:

```
QMNAME(MYQMGR)                                ROLE(Active) INSTANCE(<instance-  
name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME(london) GRPROLE(Live)  
GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) GRSTATUS(Normal)  
RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>) INITTIME(<timestamp>)  
LIVETIME(<timestamp>) ALTDATE(<date>) ALTTIME(<time>)  
GRPNAME(rome) GRPROLE(Recovery) GRPADDR(<leader-fqdn>) GRPVER(10.0.0.0) CONNGRP(yes)  
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) BACKLOG(0) INSYNC(yes)  
SYNCTIME(<timestamp>) ALTDATE(<date>) ALTTIME(<time>)
```

Note the changes to GRPNAME(rome) GRPROLE(Recovery) GRPADDR(<leader-address>) - now that the recovery group exists, the leader's fully qualified domain name is listed (<leader-address>).

- f) On the leader instance, check /var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG for the following messages to confirm the connections between the instances are secured:

```
AMQ3305I: The Native HA secure connection between the local instance and '<other-  
instance-1>' is using certificate DN 'CN=MYQMGR-REPLICATION'.  
AMQ3306I: The Native HA secure connection between the local instance and '<other-  
instance-2>' is using certificate DN 'CN=MYQMGR-REPLICATION'.
```

- g) Look for these messages in the /var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG to confirm a secure connection between Rome and London, and that the connection is ready to receive data:

```
AMQ3307I: The Native HA secure connection between the local group and 'london'  
is using certificate DN 'CN=MYQMGR-REPLICATION'.  
AMQ3280I: The Native HA group connection is ready to receive log data from  
instance '<active-instance>' in group 'london'.
```

9. Add monitoring and restarting of the queue manager.

If an instance unexpectedly fails, it is important to detect the failure and restart the instance as quickly as possible. In this example, a sample demonstrates the implementation of a 'liveness' check. The sample uses a python script and systemd daemon. When you use this method, you must use systemd to start and end the queue manager instead of the IBM MQ commands.

Follow the instructions in [“Monitoring, restarting, and ending queue manager instances” on page 484](#) to add queue manager monitoring and restarting to the delta, epsilon, and zeta hosts.

10. Add persistent messages to the queue QUEUE1 (created in [“Example: Deploying a simple Native HA configuration on Linux” on page 488](#)). The DEFPSIST (YES) attribute on the QUEUE1 queue causes messages put to the queue with the amqsputc sample to be persistent. They are replicated to the replica instances.

On the current active instance, change directory into /opt/mqm/samp/bin, which contains the sample programs. Put messages on the queue by entering the following commands:

```
./amqsput QUEUE1 MYQMGR
```

If connection to the queue manager is successful, the following response is output:

```
target queue is QUEUE1
```

Put several messages to the queue, by entering some text then pressing Enter each time. Enter a blank line to end the message input. The following message is shown:

```
Sample AMQSPUT0 end
```

Your messages are now on the queue and the command prompt is shown.

11. Perform a managed switchover, following the instructions in [“Complete a planned switchover on a Native HA CRR configuration” on page 505](#).
12. Retrieve messages sent before the switchover. On the current active instance, change directory into /opt/mqm/samp/bin, which contains the sample programs. Get the messages from the queue by entering the following command:

```
./amqsget QUEUE1 MYQMGR
```

The messages you added in step 10 are available, and are output. After a few seconds, the command exits.

What to do next

When the maintenance is complete, you can switch back to the original live group by repeating the process in steps 11.

You could also try an unplanned failover using the instructions in [“Complete an unplanned failover on a Native HA CRR configuration” on page 506](#).

If you want to learn more about how to use different certificates to secure the different parts of your CRR configuration, see [“Example: using multiple TLS certificates with Native HA CRR on Linux” on page 517](#).

Upgrading Native HA configurations

There are special considerations when you upgrade to a new version of IBM MQ on a Native HA configuration.

When planning your upgrade, you should regard the queue manager instances as together constituting a single queue manager. You should not view the instances as being independent entities. Because of this, you should aim to upgrade all your instances on their different nodes as closely together as possible. Do not adopt the cautious approach of updating one node, testing, and then updating the next node. If you proceed in this way, you will limit the availability of your queue manager.

When you upgrade instances, you affect which instances can become active until they are all at the same level again. A higher-level instance cannot write to the log of a lower-level instance. If there are two

instances with the same log data, the instance with the lower V.R.M number (that is, the older version) is favored to become the active instance. If an active instance with a higher V.R.M is elected, any remaining replicas with a lower V.R.M are not able to write new log data, and so will, over time, have a larger backlog.

Note: You can update to a new fixpack (that is, the 'F' from 'V.R.M.F') without affecting the availability of queue manager instances in the described way, but it is still recommended that you upgrade all instances at the same time.

Performance considerations when moving from circular to linear logs

If you are moving from an existing configuration that uses circular logging to a Native HA configuration, there are some performance implications that you need to consider.

Native HA uses replicated logging, which is a form of linear logging. Linear logging supports media recovery. Media recovery enables you to recreate lost or damaged objects by replaying the contents of the log, using the `rcrmqobj` command. Because of the availability of media recovery, disk requirements for circular logging are greater and performance can be affected. In particular, you should consider the following points:

- Linear logs have to accommodate media images, which are not supported in circular logs, and so linear logs are larger.
- The quantity of data written to linear logs depends on the following factors:
 - How often media images are recorded.
 - How much data there is in the queue files at the times media images are written for those queues (deep queues are not good practice, but sometimes cannot be avoided). More data in the queue files means more data must be written out to the log in media images.
 - The normal persistent messages being put/got by the applications.
- Because of the size requirements of linear logs, give attention to the following points:
 - Size the disk partition appropriately. The requirement can be significantly greater than with circular logs (depending on how much data is in the queue files at the times when their media images are being written to the logs).
 - Be aware that normal transaction/persistent messaging performance will vary, and is lower when media images are being recorded.
- Because the normal application transaction/persistent messaging work is now contending with more data being written into the log from media images, then, if the amount of data written to the log by media imaging is high, the following points might apply:
 - The applications might experience more rollbacks than with the equivalent circular log.
 - The general performance of persistent messaging is lower.

Recovering from a node failure on Native HA

If a catastrophic failure causes the loss of a node in your Native HA configuration, you can create a fresh instance to put in its place.

To recover, you delete the failed instance, and create a new, blank instance to replace it. The method for doing this depends on the platform on which you have deployed Native HA.

For all platforms, the first step is to identify the failed node. There are a variety of ways in which you can identify a failure:

- By examining error logs.
- By using the `dspmq` command.
- By using the OpenShift console (a failed instance in OpenShift is likely to be in a crashback loop).

After you have identified the failed node, proceed as follows:

In the MQ operator (OCP or AWS)

- Stop the queue manager (see [Stopping a queue manager \(mq.ibm.com/stop\)](https://mq.ibm.com/stop)) or cordon the failed instance so that it cannot be scheduled.
- Delete the persistent volume claim(s) for the corrupt or failed instance.
- Restart the queue manager using the steps described in [Stopping a queue manager \(mq.ibm.com/stop\)](https://mq.ibm.com/stop). The Operator automatically allocates a new persistent volume and recreates a blank instance.

In Kubernetes

- Cordon and evict the pods for the failed instance using the **kubectl cordon** and **kubectl drain** commands.
- Delete the persistent volume claim(s) for the corrupt or failed instance.
- Un-cordon the blank instance using **kubectl uncordon** command.

In VM or bare-metal environments

- Take the failed instance out of **systemd** control.
- Quiesce the failed instance and issue **dltmqm**.
- Recreate the instance using **crtmqm -lr**.

Remember that Native HA allows for one instance to have a catastrophic failure at a point in time without suffering any persistent data loss. If multiple instances fail concurrently, there will be a loss of service availability as the Native HA queue manager requires a majority of instances to be responsive (two out of three). If data loss occurs on a majority of instances (two out of three), there is no guarantee that the sole remaining instance will have sufficient data to be able restart the queue manager, or if it does, whether that represents the latest state.

Consider using cross-region replication to provide additional redundant replicas for a disaster recovery solution.

Comparison of Native HA CRR and Native HA IRR solutions

There are two Native HA configurations that offer disaster recovery solutions for IBM MQ: Native HA Cross-region Replication (CRR) and Native HA Inter-region replication (IRR).

Native HA CRR is available on all the platforms that support Native HA. Native HA IRR is available on Linux (virtual machines and bare metal).

A Native HA CRR configuration comprises two three-node high availability solutions, one designated as the Live system, and the other as the Recovery system. These are intended to be situated any geographical distance apart, and use asynchronous communication to replicate log data between the Live and Recovery configurations. If the active instance of a queue manager in the Live system fails, it will automatically fail over to one of the other nodes in the Live group (providing high availability functionality). If, however, you experience a complete outage of the Live site, you can manually switch over to the Recovery site, and resume running your queue manager, supplying a disaster recovery solution.

In contrast, a Native HA IRR configuration comprises two single-node systems, one designated Live and the other as Recovery. These are intended to be situated within the same general geographical area, within 100 miles (160 km) of each other, with a network latency of no more than 5 ms. Synchronous communication is used to replicate log data between the Live and Recovery configurations. A Native HA IRR configuration does not support high availability functions. If, however, you experience a complete outage of the Live site, you can manually switch over to the Recovery site, and resume running your queue manager, supplying a disaster recovery solution. You can choose whether to prioritize data consistency or service availability by setting the SyncConsistency property to Strict (the default) or Eventual.

The following lists contrast the capabilities of the two solutions:

Native HA CRR

- Fully Asynchronous replication
- Live group does not slow down if Recovery group is unable to keep pace

- Network latency between groups does not impact performance of applications connected to Live
- Prioritizes messaging availability of Live group over recovery point in an unplanned failover
- RPO ≥ 0 in an unplanned failover
- RPO of 0 in a planned failover

Native HA IRR - Strict consistency

- Synchronous replication
- Requires low latency network ($< 5\text{ms}$)
- Prioritizes ensuring an RPO of 0 in an unplanned failover
- Both Live and Recovery groups must be available for messaging service to be available
- RPO of 0 in both unplanned and planned failover

Native HA IRR - Eventual consistency

- Synchronous replication
- Requires low latency network ($< 5\text{ms}$)
- Like CRR, prioritizes messaging availability of Live group over recovery point in an unplanned failover
- If Recovery group is unavailable, Live group continues without it
- RPO ≥ 0 in an unplanned failover if replication had failed
- RPO of 0 in a planned failover

For more information on Native HA CRR configurations, see [“Native HA CRR” on page 499](#).

For more information on Native HA IRR configurations, see [“Native HA IRR” on page 522](#).

Linux Native HA CRR

A Native HA Cross-Region Replication (CRR) configuration enables you to switch the running of a queue manager to a different Native HA configuration in a different location.

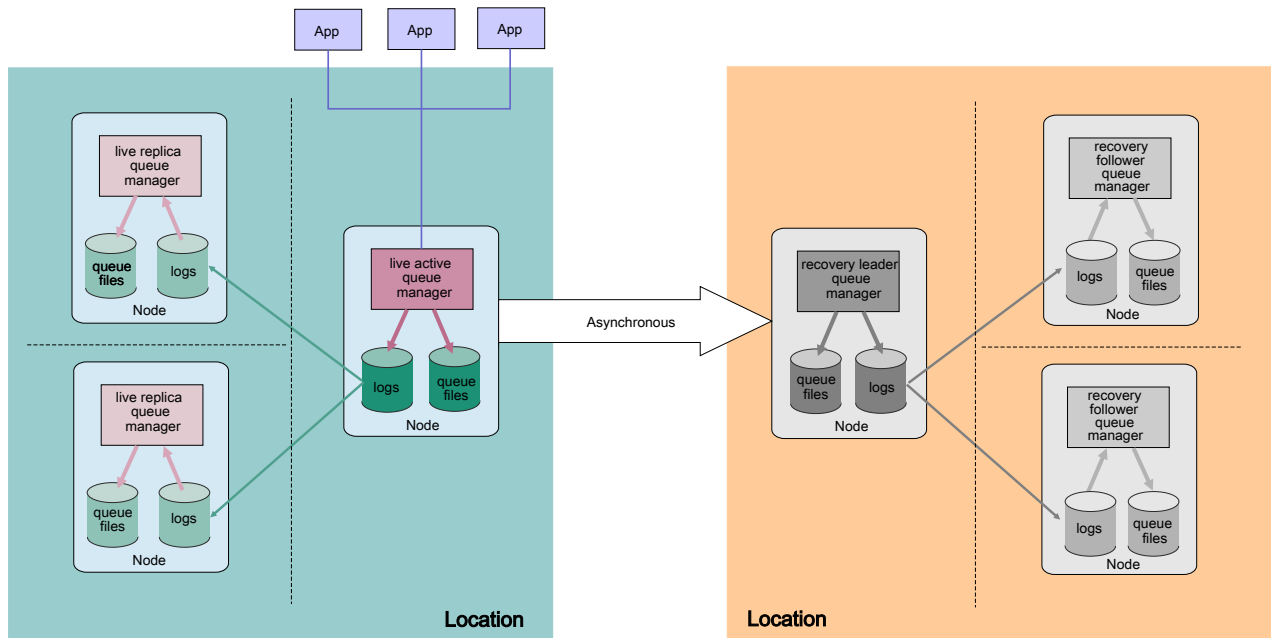
Cross-Region replication implies that the two locations are in different regions. Network latency between the two regions spans significant geographical distance and requires the use of asynchronous replication between the groups.

You can use a Native HA CRR configuration to provide a disaster recovery solution. If some disaster occurs at your live site, you can perform a failover and continue running the queue manager at the recovery site. In this situation, you cannot guarantee that the recovery logs are synchronized when the failover occurs, so there might be some loss of data. Similarly, when the failed site is restored, you might have two live instances of the queue manager and so might accrue some partitioned (split-brain) data.

You can use a Native HA CRR configuration to enable you to provide continuity while performing maintenance or testing. In this circumstance, the switchover is controlled in a way that no data loss occurs. Similarly, when you switch back there is no partitioned data to resolve.

To configure a Native HA CRR solution on Linux, a second set of three nodes using the same architecture are required to act as the recovery group. The resources of these nodes must be allocated sufficiently and uniformly so they could perform in either a Live or Recovery role, if they need to change role.

TLS encryption is mandatory when replicating log data between the two Native HA groups. It is optional, but recommended, to also configure TLS for replicating data between the members of each group.



Requested roles and Effective roles

A Requested role is the GroupRole configuration setting for a Native HA instance that is specified in the NativeHALocalInstance stanza of its `qm.ini`. The setting requests that an instance co-operate with the other instances in its group to fulfill a role. Once a majority of instances share the same Requested role, an election can take place to choose a leader of the group. On winning an election, the Requested role is compared with the Last role the instance used to derive an Effective role. See [NativeHAInstance](#) stanza of the `qm.ini` file and [NativeHALocalInstance](#) stanza of the `qm.ini` file.

Live role

A Live group is a regular Native HA queue manager consisting of three instances. One instance is elected as the active one and this instance accepts new work from applications. An existing Native HA queue manager can be migrated to a Native HA CRR configuration.

If a Requested role is not explicitly specified in the `qm.ini`, the default is considered to be Live. An elected leader of a Live group is also known as the active queue manager, this instance can accept application and tooling connections to perform new messaging work and manage queue manager objects.

Recovery role

A Recovery group also consists of three instances and one of these instances is elected as leader. The leader acts only as a proxy, accepting work sent from a paired Live group. No application connections are permitted to the Recovery group, although some control commands are allowed so that operations such as ending the queue manager instance can be completed.

The NativeHARecoveryGroup stanza in the `qm.ini` configuration file of the Live group identifies the network address for the Recovery group.

Pending role transitions

When performing a planned switchover where the current Live group becomes the new Recovery group (and the current Recovery group becomes the new Live group) there needs to be co-ordination between the groups. This co-ordination ensures that the logs are identical and achieves an RPO (recovery point objective) of zero.

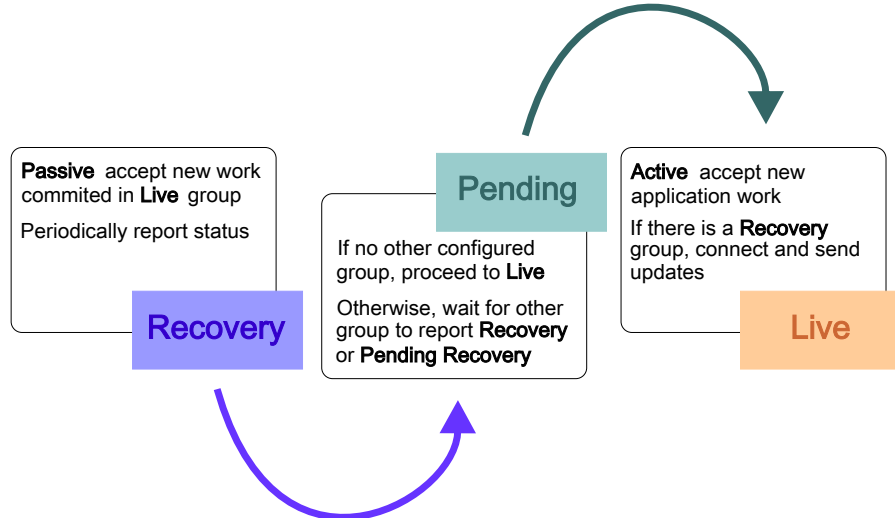
Where two groups are coordinating the switch between Recovery and Live, an effective Pending role is temporarily adopted by both groups.

Recovery role to Live role (Pending Live)

A group that has previously operated in a Recovery role that intends on changing to a Live role makes the decision on when to use the role based on whether there is another group configured.

If there is no other configured group, it transitions to the Live role immediately.

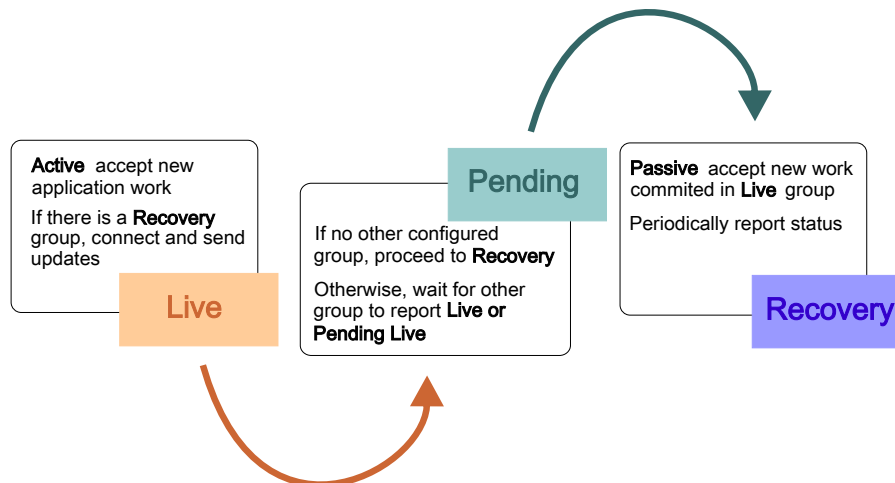
If another group is configured and enabled, the transition to a Live role is blocked until the other group adopts a Recovery or Pending Recovery role, and until both groups are in-sync.



Live role to Recovery role (Pending Recovery)

A group that had previously operated in a Live role that intends on changing to a Recovery role will make the decision on when to use the Recovery role, based on whether there is another group configured.

If another group is configured and enabled, the transition to a Recovery role is blocked until the other group adopts a Live or Pending Live role and until both groups are in-sync.



Replication and rebasing

While the two groups in a Native HA CRR configuration are connected, log updates are passed to the leader of the recovery group and passed on to the other recovery instances. This is known as **replication**.

If the network connection between the two groups is lost, it might happen that the recovery group could not catch up by using replicated logs (for example, because the live group has reused log extents that would be needed for the catch up). In this case, the recovery leader performs a **rebase**. A rebase is the process of discarding the received log data and rebuilding the queue manager from a complete set of log

data sent from the live group. Once the recovery leader has completed the rebase, the logs are replicated to the recovery instances.

If a problem is encountered during a rebase operation, then it is possible that the queue manager could be unable to start on any of the members of the recovery group. To avoid this possibility, a backup of the log is taken before the rebase operation commences. The queue manager can then be restarted using the backup log data if the rebase fails. The backup log is deleted after the rebase has succeeded.

Because of the requirement for a backup log, a Native HA CRR configuration requires at least twice the storage space of a Native HA configuration.

Instance and group naming

When configuring Native HA CRR, each set of three instances also need a collective group name. The group name is used for identification purposes (in **dspmq** output and messages), and so you might want to use a geographical name to identify the group, for example 'london' and 'rome'.

Avoid using a group name that implies the role of the group, for example 'live' and 'recovery' or 'primary' and 'backup', such names can be confusing in **dspmq** output and messages when the 'usual' roles have been swapped between the groups.

The rules for naming groups are the same as those for naming instances, see [“Instance naming” on page 483](#).

Creating the Native HA CRR solution

Create a recovery group for your Native HA CRR configuration.

About this task

These instructions assume that you have already created a Native HA group named datacenter1 comprising the hosts alpha, beta, and gamma by following the steps in [“Creating a Native HA queue manager” on page 483](#). It also assumes that you have added TLS encryption by following the steps in [“Adding TLS encryption” on page 487](#).

In this example, you add a second group named datacenter2 comprising the hosts delta, epsilon, and zeta running instances of the queue manager MYQMGR named delta, epsilon, and zeta. The set up of the second group must be identical to that of the first group in terms of platforms and resources.

You use **crtmqm** to create another three instances that work with the original three instances to form a single queue manager. It is important to ensure that the same log configuration parameters that were used to create the instances on alpha, beta, and gamma are used for the recovery group. After creation, the new group, datacenter2, rebases from the queue manager in datacenter1, and so there is no need to use **crtmqm** to create a listener.

You can view the status of a Native HA CRR queue manager by using the following methods:

- The **dspmq** command (see [dspmq \(display queue managers\)](#)).
- The IBM MQ Console (see [IBM MQ Console: Viewing queue manager status](#)).
- The IBM MQ Explorer (see [Native HA queue manager status attributes](#)).

Procedure

1. Create the first instance on the host named delta:

```
crtmqm -lr delta -lf 8192 -lp 10 -ls 10 MYQMGR
```

The queue manager is created:

```
IBM MQ queue manager 'MYQMGR' created.  
Directory '/var/mqm/qmgrs/MYQMGR' created.  
The queue manager is associated with installation 'Installation1'.  
Creating or replacing default objects for queue manager 'MYQMGR'.
```

```
Default objects statistics : 83 created. 0 replaced. 0 failed.  
Completing setup.  
Setup completed.
```

2. Create the second instance on the host named epsilon:

```
crtmqm -lr epsilon -lf 8192 -lp 10 -ls 10 MYQMGR
```

3. Create the third instance on the host named zeta:

```
crtmqm -lr zeta -lf 8192 -lp 10 -ls 10 MYQMGR
```

4. Copy the four files from the `/var/mqm/qmgrs/MYQMGR/ssl` directory on the host named alpha to the same locations on delta, epsilon, and zeta. These are the four files created in [“Adding TLS encryption”](#) on page 487, and have the following names:

- keystore.crl
- keystore.kdb
- keystore.rdb
- keystore.sth

5. In the original site, update the `qm.ini` file for each of the alpha, beta, and gamma instances that you created in [“Creating a Native HA queue manager”](#) on page 483, to add information about the group to the `NativeHALocalInstance` stanza:

```
NativeHALocalInstance:  
  CipherSpec=ANY_TLS12  
  CertificateLabel=selfsigned  
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore  
  GroupName=datacenter1  
  GroupRole=live  
  GroupLocalAddress=(9415)
```

6. Update the `qm.ini` file for each of the delta, epsilon, and zeta instances, to supply the information that enables the instances to communicate with one another, identify the TLS configuration, and set its group name and role:

```
NativeHALocalInstance:  
  CipherSpec=ANY_TLS12  
  CertificateLabel=selfsigned  
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore  
  GroupName=datacenter2  
  GroupRole=Recovery  
  GroupLocalAddress=(9415)  
  
NativeHAInstance:  
  Name=delta  
  ReplicationAddress=delta.example.com(9414)  
NativeHAInstance:  
  Name=epsilon  
  ReplicationAddress=epsilon.example.com(9414)  
NativeHAInstance:  
  Name=zeta  
  ReplicationAddress=zeta.example.com(9414)
```

7. Add information about using `datacenter1` as a recovery group, but disable it, so that it is not used unless you want to swap live and recovery roles between `datacenter1` and `datacenter2`:

```
NativeHARecoverGroup:  
  GroupName=datacenter1  
  ReplicationAddress=alpha.example.com(9415), beta.example.com(9415),  
  gamma.example.com(9415)  
  Enabled=No
```

8. Add each of the instances to `systemd` control by using the `mqmonitor.py` script and `systemd` service template (as described in [“Monitoring, restarting, and ending queue manager instances”](#) on page 484), and start the service on delta, epsilon, and zeta.

Note: It is normal for **dspmq** to show QUORUM(1/3) for a recovery group that has not yet been contacted by the live group. The recovery group first needs to replicate sufficient log data from the live group to be able to rebuild the queue manager before it shows QUORUM(3/3).

9. Reconfigure the datacenter1 group to start replicating to the datacenter2 recovery group. On each of the hosts in the datacenter1 group (alpha, beta, and gamma), edit `/var/mqm/qmgrs/MYQMGR/qm.ini` file to add recovery group information for the datacenter2 group:

```
NativeHARecoverGroup:
  GroupName=datacenter2
  ReplicationAddress=delta.example.com(9415), epsilon.example.com(9415),
  zeta.example.com(9415)
  Enabled=Yes
```

The comma-separated list of possible group leaders enables clients or channels to find an active queue manager instance.

10. Restart all the instances by requesting a service restart:

```
sudo systemctl restart mqmonitor@MYQMGR
```

11. After a new active instance is elected in the datacenter1 group, it will attempt to connect to the datacenter2 recovery group identified in the `qm.ini` file. You can use the **dspmq** command on any member in the datacenter1 or datacenter2 group to view the status of the two groups:

```
dspmq -m MYQMGR -o nativeha -g
```

Both groups should report QUORUM(3/3) and indicate that the connection to the other group is currently connected. For example, the status from alpha in the datacenter1 group might be:

```
QMNAME(MYQMGR)                                ROLE(Active) INSTANCE(alpha)
INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:20:26799>) GRPNAME(datacenter1) GRPROLE(Live)
HAINITDA(2026-07-08) HAINITL(<0:0:11:1226>) HAINITTI(13.27.18)
  GRPNAME(datacenter1) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) GRSTATUS(Normal)
RCOVLSN(<0:0:20:26799>) RCOVTIME(2026-07-08T12:28:18.969864Z) INITLSN(<0:0:11:1226>)
INITTIME(2026-07-08T12:27:18.967470Z) LIVETIME(2026-07-08T12:27:51.416974Z)
ALTDATA(2026-07-08) ALTTIME(13.28.27)
  GRPNAME(datacenter2) GRPROLE(Recovery) GRPADDR(delta) GRPVER(10.0.0.0) CONNGRP(yes)
GRSTATUS(Normal) RCOVLSN(<0:0:20:26799>) RCOVTIME(2026-07-08T12:28:18.969864Z) BACKLOG(0)
INSYNC(yes) SYNCTIME(2026-07-08T12:28:41.978393Z) ALTDATA(2026-07-08) ALTTIME(13.28.35)
```

The status from delta in the datacenter2 group might be:

```
QMNAME(MYQMGR)                                ROLE(Leader) INSTANCE(delta)
INSYNC(yes) QUORUM(3/3) GRPLSN(<0:0:20:38689>) GRPNAME(datacenter2) GRPROLE(Recovery)
HAINITDA() HAINITL() HAINITTI()
  GRPNAME(datacenter2) GRPROLE(Recovery) GRPADDR(delta) GRPVER(10.0.0.0) GRSTATUS(Normal)
RCOVLSN(<0:0:20:38689>) RCOVTIME(2026-07-08T12:28:55.594739Z) BACKLOG(0) INSYNC(yes)
SYNCTIME(2026-07-08T12:31:34.645479Z) ALTDATA(2026-07-08) ALTTIME(13.29.06)
  GRPNAME(datacenter1) GRPROLE(Live) GRPADDR(Unknown)
GRPVER(10.0.0.0) CONNGRP(yes) GRSTATUS(Normal)
RCOVLSN(<0:0:20:38689>) RCOVTIME(2026-07-08T12:28:55.594739Z) INITLSN(<0:0:11:1226>)
INITTIME(2026-07-08T12:27:18.967470Z) LIVETIME(2026-07-08T12:27:51.416974Z)
ALTDATA(2026-07-08) ALTTIME(13.31.35)
```

The statuses show that datacenter1 is live and datacenter2 is recovery, and indicates how closely the recovery group is tracking the live workload.

What to do next

After you have configured Native HA CRR, you can perform a planned switchover if you need to apply maintenance to your live group, or an unplanned failover if your live group fails for some reason. See [“Complete a planned switchover on a Native HA CRR configuration” on page 505](#) and [“Complete an unplanned failover on a Native HA CRR configuration” on page 506](#).

Complete a planned switchover on a Native HA CRR configuration

You can perform a planned switchover between the live and recovery groups in a Native HA Cross-Region Replication (CRR) configuration.

About this task

You might require to switch from your live group to your recovery group if you want to test your configuration, or need to apply maintenance. (If you upgrading IBM MQ, see [“Upgrading Native HA configurations”](#) on page 496 for guidance.)

When you perform a planned switchover, the system takes steps to ensure that you do not lose data or encounter a partitioned (split-brain) problem. You request the switchover by changing the group role on each of the groups. For a while both groups enter a pending status while their recovery logs are synchronized. When this is achieved, the groups adopt their new roles, a new leader is elected in the new live group and takes the active role and runs the queue manager.

This task uses the example configuration created in [“Creating the Native HA CRR solution”](#) on page 502.

Procedure

1. Specify that the recovery group, datacenter2, should become the live group by editing the qm.ini file for each queue manager instance in the group (delta, epsilon, and zeta). Change the GroupRole to Live, and ensure NativeHARecoveryGroup Enabled is Yes.

```
NativeHALocalInstance:
  .
  .
  .
  GroupName=datacenter2
  GroupRole=Live

NativeHARecoveryGroup:
  GroupName=datacenter1
  .
  .
  .
  Enabled=Yes
```

2. Restart each instance in the datacenter2 group:

```
sudo systemctl restart mqmonitor@MYQMGR
```

After the restart and election of a leader, the **dspmq -m MYQMGR -o nativeha -g** command should confirm the pending Live role. When the datacenter2 group becomes live it handles the queue manager work, and testing or maintenance can be carried out on the nodes of the datacenter1 group.

Note: While the instance is in the pending Live role, you can change your mind about the switchover and revert to the previous Live and Recovery arrangement. Reverse the changes that you made in steps 1 and 3 and restart the instances.

3. Specify that the live group, datacenter1, should become the recovery group by editing the qm.ini file for each queue manager instance in the group (alpha, beta, and gamma). Change the GroupRole to Recovery, and ensure NativeHARecoveryGroup Enabled is Yes.

```
NativeHALocalInstance:
  .
  .
  .
  GroupName=datacenter1
  GroupRole=Recovery

NativeHARecoveryGroup:
  GroupName=datacenter2
  .
  .
  .
  Enabled=Yes
```

Changing a group role from Live to Recovery, or Recovery to Live with this configuration instructs each group to co-ordinate its role transition with the other. While the negotiation is taking place, each group adopts a pending role which indicates the role it is in the process of changing to.

4. Restart each instance in the datacenter1 group:

```
sudo systemctl restart mqmonitor@MYQMGR
```

After the restart and election of a leader, the **dspmq -m MYQMGR -o nativeha -g** command should confirm the pending Recovery role.

What to do next

When the maintenance is complete, you can switch back to the original live group by reversing the process, restoring datacenter1 as the live group and datacenter2 as the recovery group.

Complete an unplanned failover on a Native HA CRR configuration

You perform an unplanned failover between the live and recovery groups in a Native HA Cross-Region Replication (CRR) configuration when some disaster occurs at the live site and it can no longer operate.

About this task

This task uses the example configuration created in [“Creating the Native HA CRR solution” on page 502](#).

When the live group goes down, the recovery group cannot detect this as it simply stops receiving replicated log data, which could just mean there is no ongoing workload. An unplanned failover must therefore be manually carried out by changing the recovery group's role to live and notifying it that the previously live group is disabled so that it does not attempt to replicate data to it.

The recovery group might not be in-sync with the live group as log data is replicated asynchronously. If the recovery group is not in-sync with the live group, data is lost when an unplanned failover occurs. Before initiating an unplanned failover, issue the **dspmq -o nativeha -g** command on the recovery group, and use the RCOVLSN and RCOVTIME values to view the log sequence number (LSN) and time that the group can recover to.

In this example, the datacenter1 groups fails, and the datacenter2 group assumes the live role.

Procedure

1. Ensure that applications have disconnected from the datacenter1 group, and are configured to reconnect to the datacenter2 group (which is currently the recovery group).
2. Stop each of the instances in the datacenter1 group to mimic losing the nodes in datacenter1:

```
sudo systemctl stop mqmonitor@MYQMGR
```

3. Modify the `/var/mqm/qmgrs/MYQMGR/qm.ini` file on each instance in the datacenter2 group (delta, epsilon, and zeta), changing the group role from Recovery to Live:

```
NativeHALocalInstance:
  GroupName=datacenter2
  GroupRole=Live
  Enabled=No
```

Because the NativeHARecoveryGroup stanza in the datacenter2 group configuration is disabled (NativeHARecoveryGroup Enabled is set to No), the group processes this change as an unplanned failover and does not attempt to co-ordinate the change with the datacenter1 group.

4. Restart each of the instances in the datacenter2 group:

```
sudo systemctl restart mqmonitor@MYQMGR
```

If you issue a **dspmq -o nativeha -g** command for the queue manager MYQMGR after each instance in the group has restarted will see a GRPROLE of Live. An election is held and one of the instances becomes the group leader and the active instance and handles the queue manager's work.

5. To restart replication after the failover, you can either use the original live group as the recovery group, or create a new recovery group at a new site.
 - If the original live site is restored, before restarting the original live group, change the role of the group to recovery. This ensures that the group does not resume its live role.
 - If the original live site cannot be restored, create a new Native HA recovery group at a new site.
6. If the configuration on the original live site is not changed, and the group resumes its live role when it is restored, you will have two active instances of the queue manager. If this is the case, the partitioned state is detected and no more replication occurs between the two groups. You must resolve which data to keep before you restore the original live and recovery roles. See [“Resolving a partitioned \(split brain\) problem”](#) on page 510.

Swapping live and recovery roles after an unplanned failure

After an unplanned failure has occurred and you have failed over to your recovery group, you might want to reconfigure so that your original live group is restored as the new recovery group.

About this task

In the example configuration described in [“Creating the Native HA CRR solution”](#) on page 502, datacenter1 is the live group and datacenter2 is the recovery group. If there was a major power outage at the datacenter1 site, for example, you would switch over to the datacenter2 site using the process described in [“Complete an unplanned failover on a Native HA CRR configuration”](#) on page 506. When power is restored to the datacenter1 site, you might decide to keep datacenter2 as the live group, and make datacenter1 the recovery group. There are some special considerations when you take this course of action.

The decision about which site will be live and which will be recovery after an unplanned failover must take into consideration the state of the data on each of the sites, because log data and queue manager data on the new recovery site will be deleted and overwritten with data from the live site when you reconfigure.

If you decide to swap roles so that your original live group becomes the new recovery group, there are two alternative procedures that you can follow, both are completed on the original live configuration, now to become the recovery (datacenter1 in the example):

- Delete and recreate the queue manager - data is deleted and replaced when the new live group connects.
- Change the queue manager .ini file and restart - when the live group connects a rebase occurs and data is replaced.

In both cases, you also need to change the configuration of the original live group to identify its new recovery role.

Procedure

- To swap roles by deleting and recreating the queue manager:
 - a) Delete the queue manager on every node of the previous live group (in the example, the queue manager is called MYQMGR):

```
endmqm MYQMGR
dltmqm MYQMGR
```

All the log and queue manager data for this instance of MYQMGR are deleted.

- b) Recreate the queue manager on each node in the group, specifying that it is a Native HA CRR queue manager with replicated logging (in the example, the node name is alpha):

```
crtmqm -lr alpha -lf 8192 -lp 10 -ls 10 MYQMGR
```

- c) Specify that each instance of the queue manager in this group is in the recovery role by editing the `qm.ini` file on each node (in the example, the node name is alpha):

```
NativeHALocalInstance:
  Name=alpha
  GroupName=datacenter1
  KeyRepository=/var/mqm/qmgrs/MYQM/ssl/key
  CertificateLabel=MyCertificate
  CipherSpec=NULL
  GroupRole=Recovery
  GroupCipherSpec=ANY_TLS12
  GroupLocalAddress=(4453)

NativeHAInstance:
  Name=alpha
  ReplicationAddress=alpha.example.com(4453)
NativeHAInstance:
  Name=beta
  ReplicationAddress=beta.example.com(4453)
NativeHAInstance:
  Name=gamma
  ReplicationAddress=gamma.example.com(4453)
```

Repeat for the other two nodes in the group.

- d) Ensure that the queue manager instances on the new live site (datacenter2) are correctly configured to have datacenter1 as its recovery group. For example, for the datacenter2 node named delta:

```
NativeHALocalInstance:
  Name=delta
  GroupName=datacenter2
  KeyRepository=/var/mqm/qmgrs/MYQM/ssl/key
  CertificateLabel=MyCertificate
  CipherSpec=NULL
  GroupRole=Live
  GroupCipherSpec=ANY_TLS12
  GroupLocalAddress=(4453)

NativeHARecoveryGroup:
  GroupName=datacenter1
  ReplicationAddress=alpha.example.ibm.com(4453), beta.example.com(4453),
  gamma.example.com(4453)
```

- e) You can now start the queue manager instance in the group datacenter1 by issuing **strmqm** on each of the three nodes :

```
strmqm MYQMGR
```

Data is replicated to MYQMGR on datacenter1 from the live group datacenter2.

- To swap roles by changing the queue manager `qm.ini` file:
 - a) Specify that the instance of the queue manager MYQMGR on datacenter1 is in the recovery role by editing the `qm.ini` file on each node in the group. For example, on node alpha:

```
NativeHALocalInstance:
  Name=alpha
  GroupName=datacenter1
  KeyRepository=/var/mqm/qmgrs/MYQM/ssl/key
  CertificateLabel=MyCertificate
  CipherSpec=NULL
  GroupRole=Recovery
  GroupCipherSpec=ANY_TLS12
  GroupLocalAddress=(4453)
```

- b) Ensure that the queue manager instances in the new live group (datacenter2) are correctly configured to have datacenter1 as their recovery group. For example, for the node named delta:

```
NativeHALocalInstance:
  Name=delta
  GroupName=datacenter2
  KeyRepository=/var/mqm/qmgrs/MYQM/ssl/key
  CertificateLabel=MyCertificate
```



```
CipherSpec=NULL
GroupRole=Live
GroupCipherSpec=ANY_TLS12
GroupLocalAddress=(4453)

NativeHARecoverGroup:
  GroupName=datacenter1
  ReplicationAddress=alpha.example.ibm.com(4453), beta.example.com(4453),
  gamma.example.com(4453)
```

- c) You can now start the queue manager on the nodes in the datacenter1 group:

```
startmq MYQMGR
```

The live leader instance in the datacenter2 group performs a rebase to bring the instances in the datacenter1 group into sync.

Upgrading Native HA CRR configurations

How to upgrade your Native HA CRR configuration to a new version of IBM MQ.

About this task

When planning your upgrade, you should regard the queue manager instances as together constituting a single queue manager. You should not view the instances as being independent entities. Because of this, you should aim to upgrade all your instances in a group as closely together as possible. Do not adopt the cautious approach of updating one node, testing, and then updating the next node. If you proceed in this way, you will limit the availability of your queue manager.

In a Native HA CRR configuration, all instances in the group performing the recovery role must be at an IBM MQ version equal to or higher than the group performing the live role. You cannot fail over or switch over to a group that is a lower version. Because of this, you should apply upgrades to all members of the recovery group before applying it to all members of the live group. The upgrade of IBM MQ is only complete when all instances of both the recovery and live groups have been upgraded.

Note: You can update to a new fixpack (that is, the 'F' from 'V.R.M.F') without affecting the availability of queue manager instances in the described way, but it is still recommended that you upgrade all instances at the same time.

There are two approaches to upgrading your Native HA CRR configuration, depending on how much down time you can afford:

- You can upgrade all your recovery group instances, then upgrade your live group instances. You do not perform any switch over, so the queue manager is unavailable during upgrade.
- You can upgrade your recovery group instances, then switch over to this group while you upgrade your previous live group. You can switch back on completion of the upgrade. The queue manager is only unavailable while you perform the switch overs.

Procedure

- To upgrade without performing a switch over:
 - a) On the recovery group, stop your queue managers, for example:

```
endmqm MYQMGR
```

- b) Upgrade all the nodes in your recovery group to the new version of IBM MQ. See [Upgrading an IBM MQ installation on Linux](#).
- c) On the live group, stop your queue managers.
- d) Upgrade all the nodes in your live group to the new version of IBM MQ.
- e) Restart your queue managers on the recovery group. For example:

```
startmqm MYQMGR
```

- f) Restart your queue managers on the live group.
- To upgrade with a planned switchover:
 - a) On the recovery group, stop your queue managers, for example:

```
endmqm MYQMGR
```

- b) Upgrade all the nodes in your recovery group to the new version of IBM MQ. See [Upgrading an IBM MQ installation on Linux](#).
- c) Complete a switchover to the recovery group that you have just upgraded. See [“Complete a planned switchover on a Native HA CRR configuration”](#) on page 505.
- d) Upgrade all the nodes in your original live group to the new version of IBM MQ.
- e) Switch back to your original live group.

Resolving a partitioned (split brain) problem

A partitioned, or 'split-brain', condition can occur if a queue manager in a Native HA Cross-Region Replication (CRR) configuration runs in both the HA groups at the same time.

The condition can arise when an unplanned failover occurs. Communication with the Live group is lost and so you make the queue manager the primary on the Recovery group and it starts running. Meanwhile, the queue manager is still running on the original Live group. There is a possibility that the old Live group is still able to update its log and this creates an unresolvable branch in the data maintained by both two groups. You must discard the data from one of the groups before they can be rejoined.

To resolve the partitioned condition, you choose which group data to retain and which to delete. It might be that the groups were not in sync when the unplanned failover was triggered, or it might be that application workload was able to be processed by both groups. The definition of "best" data is largely subjective. For example, while one group might have stored more data in its log, this might be largely automatically recorded media images of IBM MQ objects; whereas the other group might have processed more valuable business transactions. Whichever group is chosen, it is likely that, in the recovery from the partitioned condition, the unreplicated message and transaction outcomes on the group to be deleted will be useful when reconciling the possible data loss.

IBM MQ detects a partitioned condition when a group with a Live role connects to a another group that is running with a Live role. If you issue the command `dspmq -o nativeha -g` in both groups, the status `GRSTATUS(Partitioned)` is returned. The replication of log data is suspended between both groups. The Partitioned state continues to be reported until the group is able to replicate sufficient log data to another group with a Recovery role.

If it is not an obvious choice which group to retain and, if log extents are still available from both groups at the time of the unplanned failover, you can compare the log contents.

Use the `dspmq -o nativeha -g` command to identify the LSN (log sequence number) of the exact point in the log that a group became Live. The LSN is returned in the form `INITLSN (nnnnn:nnnnn:nnnnn:nnnnn)`. The time of this event is also reported by the `INITTIME` attribute. Then use the `dmpmqlog -s startLSN` command to examine the contents of a log from that point. Use `dspmq` and `dmpmqlog` in combination to identify log records that were written by one group and not replicated to the other. The `dmpmqlog` command also reports a summary of the messaging and transaction activity, which can be useful when making the choice.

Example

The groups `datacenter1` and `datacenter2` were configured initially in a Live and Recovery configuration. A decision to convert the `datacenter2` group from Recovery to Live was made and an unplanned failover occurred. However the `datacenter1` group remained able to progress some application workload which was not replicated to `datacenter2`. Meanwhile, some applications connected to `datacenter2` and put some

messaging workload through. The INITLSN from the datacenter2 group indicates the point in the log that the datacenter2 group branched away from the content of the datacenter1 group.

A copy of the queue manager data is needed for the comparison process and the queue manager needs to be inactive.

After the analysis of the log data has been completed and the choice of which group data to discard is made, you can discard it by changing the group role to Recovery and either deleting all of the queue manager persistent volumes, or removing the Recovery group information.

Example: Deploying a simple Native HA CRR configuration on Linux

This example installs a queue manager using a simple Native HA Cross-Region Replication (CRR) configuration across six Linux hosts.

Before you begin

To complete this example, you must first complete the following tasks:

- Follow the steps in [Checking requirements on Linux](#).
- Follow the steps in [Planning to install IBM MQ on Linux](#)
- For each of the six Linux hosts, follow the steps in [Preparing the system on Linux](#).
- Download the 10.0.0 or above IBM MQ base install images for your Linux distribution (see [IBM MQ downloads](#)).

About this task

The Native HA CRR configuration defines two Native HA groups, which together form a highly available queue manager. Usually these groups are created in two different regions to provide the cross-region replication. However, there are no constraints on where groups can be located and, for this example, it is easier to create both groups on Linux hosts in a single location. For more information about Native HA CRR configurations on Linux, see [“Native HA CRR” on page 499](#).

This example defines a London group and a Rome group, each consisting of three IBM MQ instances installed on their own Linux host. The host names of the London instances are alpha, beta, and gamma and those in the Rome group are delta, epsilon, and zeta. Throughout the example, replace these names with the host names of your systems. For simplicity, this example uses a single keystore to secure all the queue manager instance and group traffic. See [“Example: using multiple TLS certificates with Native HA CRR on Linux” on page 517](#) for more information on how certificates are used in Native HA CRR, and guidance on using different certificates to secure group and instance traffic.

This example walks through defining two Native HA groups on Linux hosts and providing cross-region replication, setting up monitoring and restarts, and performing a switchover.

This example uses the default installation location (/opt/mqm) and data directory (/var/mqm). If you are using non-default locations, be sure to replace these paths accordingly throughout the example.

Procedure

1. Install IBM MQ on all six Linux hosts.

Follow the instructions in [Installing and uninstalling IBM MQ on Linux](#) to accept the IBM MQ license and install the IBM MQ server on all six of your hosts.

For this example, it is easiest to use the default locations for installation (/opt/mqm) and data (/var/mqm). If you change the install or data location on one system, ensure that you mirror the same change across all other hosts.

2. Set up environments ready to create the Native HA instances:

- a) On each Linux host, log in as the mqm user and set up your environment by using the following command:

```
. /opt/mqm/bin/setmqenv -s
```

- b) Check that the environment is set up correctly by entering the following command:

```
dspmqver
```

If the command completes successfully, and the expected version number and installation name are returned, the environment is set up correctly.

3. Create the London queue manager instances. On hosts alpha, beta, and gamma create a queue manager with log replication. (You might want to use the AutoConfig stanza to ensure objects are always defined in a uniform way, the **-ic** option can be added to **crtmqm** to supply a mqsc file for the queue manager.)

- a) Create the first London instance on the node named alpha:

```
crtmqm -lr alpha -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

The queue manager is created:

```
IBM MQ queue manager 'MYQMGR' created.  
Directory '/var/mqm/qmgrs/MYQMGR' created.  
The queue manager is associated with installation 'Installation1'.  
Creating or replacing default objects for queue manager 'MYQMGR'.  
Default objects statistics : 83 created. 0 replaced. 0 failed.  
Completing setup.  
Setup completed.
```

- b) Create the second London instance on the node named beta:

```
crtmqm -lr beta -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

- c) Create the third London instance on the node named gamma:

```
crtmqm -lr gamma -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

4. Create the Rome queue manager instances. As the mqm user, on hosts delta, epsilon, and zeta create a queue manager with log replication. (If you used the AutoConfig stanza for the London instances, use the **crtmqm -ic** option for these instances as well.)

- a) Create the first Rome instance on the node named delta:

```
crtmqm -lr delta -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

- b) Create the second Rome instance on the node named epsilon:

```
crtmqm -lr epsilon -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

- c) Create the third Rome instance on the node named zeta:

```
crtmqm -lr zeta -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

5. Create a keystore for securing the instance and group traffic. In this example, we use the same keystore for both traffic between the instances and the groups.

- a) To create this keystore, log on to the alpha host and run the following commands:

```
runmqakm -keydb -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -stash  
runmqakm -cert -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label  
nha-qm-replication -dn CN=MYQMGR-REPLICATION -size 2048
```

- b) Copy the four keystore files to the same location (`/var/mqm/qmgrs/MYQMGR/ssl/`) on the beta, gamma, delta, epsilon, and zeta hosts, for example, by using `sftp`.

- c) On each host (alpha, beta, gamma, delta, epsilon, and zeta), ensure that the files have mqm group ownership and have read access for the mqm group:

```
chown :mqm /var/mqm/qmgrs/MYQMGR/ssl/keystore.*
chmod g+r /var/mqm/qmgrs/MYQMGR/ssl/keystore.*
```

Important: The examples described here are not suitable for a production environment, and are solely intended as examples to get going quickly. Certificate management is a complex subject for advanced users. For production, you must consider things like rotation, revocation, key length, disaster recovery and much more.

6. Create the London group. The alpha, beta, and gamma hosts form the London group, with the initial role of Live. Later, the delta, epsilon, and zeta hosts will form the Rome group, and its initial role will be Recovery. The groups are created by editing the `/var/mqm/qmgrs/MYQMGR/qm.ini` files for each queue manager instance.
 - a) Update the `NativeHALocalInstance` stanza for each of the alpha, beta, and gamma instances to define the group and the certificate to use to secure the instance traffic:

```
NativeHALocalInstance:
  Name=<instance-name>
  GroupName=london
  GroupRole=Live
  GroupLocalAddress=(9415)
  CipherSpec=ANY_TLS12
  CertificateLabel=nha-qm-replication
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore
```

Where:

- `<instance-name>` is the name of the instance that you are configuring (that is, alpha, beta, or gamma).
 - `CertificateLabel` specifies the certificate for traffic between instances. In this example, the same certificate is used for the traffic between instances and groups. (The `GroupCertificateLabel` specifies the certificate from group to group traffic. When it is not specified, the groups use the same certificate as the instance traffic.)
- b) For each of the alpha, beta, and gamma instances add the local group and recovery group members:

```
NativeHAInstance:
  Name=alpha
  ReplicationAddress=alpha.example.com(9414)
NativeHAInstance:
  Name=beta
  ReplicationAddress=beta.example.com(9414)
NativeHAInstance:
  Name=gamma
  ReplicationAddress=gamma.example.com(9414)
NativeHARecoverGroup:
  GroupName=rome

ReplicationAddress=delta.example.com(9415),epsilon.example.com(9415),zeta.example.com(9415)
Enabled=Yes
```

Where:

- The `NativeHAInstance` stanzas define all the instances in the London Native HA group
 - The `NativeHARecoverGroup` stanza provides details about the recovery group and its members
- (Replace host names and replication addresses with the names and replication addresses of your own Linux nodes.)
- c) Start the instances in the London group. You can now start the queue manager instances on alpha, beta, and gamma hosts:

```
startmqm MYQMGR
```

- d) Verify the London group has deployed correctly. Issue the **dspmq -o nativeha -g** command on the alpha, beta, and gamma hosts, to display the group status. On the active instance, the status is:

```
QMNAME(MYQMGR)                                ROLE(Active) INSTANCE(<instance-  
name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME(london) GRPROLE(Live)  
GRPNAME(London) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) GRSTATUS(Normal)  
RCOVLN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>) INITTIME(<timestamp>)  
LIVETIME(<timestamp>) ALTDAT(<date>) ALTTIME(<time>)  
GRPNAME(rome) GRPROLE(Unknown) GRPADDR(Unknown) GRPVER(?.?.?.?) CONNGRP(no)  
GRSTATUS(Unknown) RCOVLN() RCOVTIME() ALTDAT() ALTTIME()
```

The key parts of the status are:

- **ROLE(Active)**
- **GRPNAME(london) GRPROLE(Live)** - London is the live group
- **QUORUM(3/3)** - it has full quorum, note this can take a few seconds
- **GRPNAME(rome)** - the recovery group will be in Rome. (The GRPADDR is unknown as the Rome group does not exist yet.)

Note which host in the London group has the Active role. The other hosts will have a similar status, but the **ROLE** is **Replica**.

- e) On the active instance, check `/var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG` for the following messages to confirm the connections between the instances are secured:

```
AMQ3305I: The Native HA secure connection between the local instance and '<other-  
instance-1>' is using certificate DN 'CN=MYQMGR-REPLICATION'.  
AMQ3305I: The Native HA secure connection between the local instance and '<other-  
instance-2>' is using certificate DN 'CN=MYQMGR-REPLICATION'.
```

7. Create the Rome group on the delta, epsilon, and zeta hosts.

- a) Edit the `/var/mqm/qmgrs/MYQMGR/qm.ini` file for each host. Add the following to the **NativeHALocalInstance** stanza:

```
NativeHALocalInstance:  
  Name=<instance-name>  
  GroupName=rome  
  GroupRole=Recovery  
  GroupLocalAddress=(9415)  
  CipherSpec=ANY_TLS12  
  CertificateLabel=nha-qm-replication  
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore
```

- **<instance-name>** is the name of the instance that you are configuring (that is, delta, epsilon, or zeta).

- b) For each of the delta, epsilon, and zeta hosts add the local group and recovery group members:

```
NativeHAInstance:  
  Name=delta  
  ReplicationAddress=delta.example.com(9414)  
NativeHAInstance:  
  Name=epsilon  
  ReplicationAddress=epsilon.example.com(9414)  
NativeHAInstance:  
  Name=zeta  
  ReplicationAddress=zeta.example.com(9414)  
NativeHARecoverGroup:  
  GroupName=london  
  
ReplicationAddress=alpha.example.com(9415),beta.example.com(9415),gamma.example.com(9415)  
Enabled=Yes
```

Where:

- The **NativeHAInstance** stanzas define all the instances in the Rome Native HA group.
- The **NativeHARecoverGroup** stanza provides details about the live group and its members.

(Replace host names and replications addresses with the names and replication addresses of your own Linux nodes.)

- c) Start the instances in the Rome group. You can now start the queue manager instances on delta, epsilon, and zeta hosts:

```
stimqm MYQMGR
```

- d) Verify the Rome group has deployed correctly. Issue the **dspmq -o nativeha -g** command on the delta, epsilon, and zeta instances, to get their group status. On the leader, the status is:

```
QMNAME(MYQMGR)                                ROLE(Leader) INSTANCE(<instance-  
name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME(rome) GRPROLE(Recovery)  
GRPNAME(rome) GRPROLE(Recovery) GRPADDR(<leader-fqdn>) GRPVER(10.0.0.0)  
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) BACKLOG(0) INSYNC(yes)  
SYNCTIME(<timestamp>) ALTDATETIME(<date>) ALTTIME(<time>)  
GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) CONNGRP(yes)  
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>)  
INITTIME(<timestamp>) LIVETIME(<timestamp>) ALTDATETIME(<date>) ALTTIME(<time>)
```

It might take a few seconds to reach full quorum. The key parts of this status are:

- **ROLE(Leader)**
- **GRPNAME(rome) GRPROLE(Recovery)** - Rome is the recovery group
- **QUORUM(3/3)** - it has full quorum, note this can take a few seconds
- **GRPNAME(london)** - London is the live group

- e) Issue the **dspmq -o nativeha -g** command on the active node of the London group, the status is now:

```
QMNAME(MYQMGR)                                ROLE(Active) INSTANCE(<instance-  
name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME(london) GRPROLE(Live)  
GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) GRSTATUS(Normal)  
RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>) INITTIME(<timestamp>)  
LIVETIME(<timestamp>) ALTDATETIME(<date>) ALTTIME(<time>)  
GRPNAME(rome) GRPROLE(Recovery) GRPADDR(<leader-fqdn>) GRPVER(10.0.0.0) CONNGRP(yes)  
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) BACKLOG(0) INSYNC(yes)  
SYNCTIME(<timestamp>) ALTDATETIME(<date>) ALTTIME(<time>)
```

Note the changes to **GRPNAME(rome) GRPROLE(Recovery) GRPADDR(<leader-address>)** - now that the recovery group exists, the leader's fully qualified domain name is listed (**<leader-address>**).

- f) On the leader instance, check **/var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG** for the following messages to confirm the connections between the instances are secured:

```
AMQ3305I: The Native HA secure connection between the local instance and '<other-  
instance-1>' is using certificate DN 'CN=MYQMGR-REPLICATION'.  
AMQ3306I: The Native HA secure connection between the local instance and '<other-  
instance-2>' is using certificate DN 'CN=MYQMGR-REPLICATION'.
```

- g) Look for these messages in the **/var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG** to confirm a secure connection between Rome and London, and that the connection is ready to receive data:

```
AMQ3307I: The Native HA secure connection between the local group and 'london'  
is using certificate DN 'CN=MYQMGR-REPLICATION'.  
AMQ3280I: The Native HA group connection is ready to receive log data from  
instance '<active-instance>' in group 'london'.
```

8. Add monitoring and restarting of the queue manager.

If an instance unexpectedly fails, it is important to detect the failure and restart the instance as quickly as possible. In this example, a sample demonstrates the implementation of a 'liveness' check. The sample uses a python script and systemd daemon. When you use this method, you must use systemd to start and end the queue manager instead of the IBM MQ commands.

Follow the instructions in [“Monitoring, restarting, and ending queue manager instances” on page 484](#) to add queue manager monitoring and restarting to the alpha, beta, gamma, delta, epsilon, and zeta hosts.

9. Define a queue. On the active instance, start MQSC by entering the following command:

```
runmqsc MYQMGR
```

A message indicates when MQSC starts. MQSC has no command prompt. Define a local persistent queue:

```
DEFINE QLOCAL('QUEUE1') DEFPSIST(YES)
```

A message indicates when the queue is created. Stop MQSC by entering the following command:

```
end
```

Messages are shown, followed by the command prompt. The QUEUE1 queue is replicated to the other instances in both the London and Rome groups.

10. Add persistent messages to the queue. The DEFPSIST(YES) attribute on the QUEUE1 queue causes messages put to the queue with the amqsputc sample to be persistent. They are replicated to the recovery group.

Change directory into /opt/mqm/samp/bin, which contains the sample programs. Put messages on the queue by entering the following commands:

```
./amqsput QUEUE1 MYQMGR
```

If connection to the queue manager is successful, the following response is output:

```
target queue is QUEUE1
```

Put several messages to the queue, by entering some text then pressing Enter each time. Enter a blank line to end the message input. The following message is shown:

```
Sample AMQSPUT0 end
```

Your messages are now on the queue and the command prompt is shown.

11. Perform a managed switchover, following the instructions in [“Complete a planned switchover on a Native HA CRR configuration”](#) on page 505.
12. Retrieve messages sent before the switchover. On the current active instance, change directory into /opt/mqm/samp/bin, which contains the sample programs. Get the messages from the queue by entering the following command:

```
./amqsget QUEUE1 MYQMGR
```

The messages you added in step 10 are available, and are output. After a few seconds, the command exits.

What to do next

When the maintenance is complete, you can switch back to the original live group by repeating the process in steps 11.

You could also try an unplanned failover using the instructions in [“Complete an unplanned failover on a Native HA CRR configuration”](#) on page 506.

If you want to learn more about how to use different certificates to secure the different parts of your CRR configuration, see [“Example: using multiple TLS certificates with Native HA CRR on Linux”](#) on page 517.

Example: using multiple TLS certificates with Native HA CRR on

This example describes how to use multiple certificates to secure traffic in a Native HA CRR configuration.

About this task

This example starts with an overview of how TLS and certificates are used in a Native HA CRR configuration, then uses simple steps to demonstrate how `nhacrr-london-group`, `nhacrr-rome-group`, `nhacrr-london-crr` and `nhacrr-rome-crr` traffic is secured.

The steps in the procedure section below walk you through altering the [“Example: Deploying a simple Native HA CRR configuration on Linux”](#) on [page 511](#) deployment to use different certificates for each type of traffic, and change which CipherSpec is used for group to group traffic.

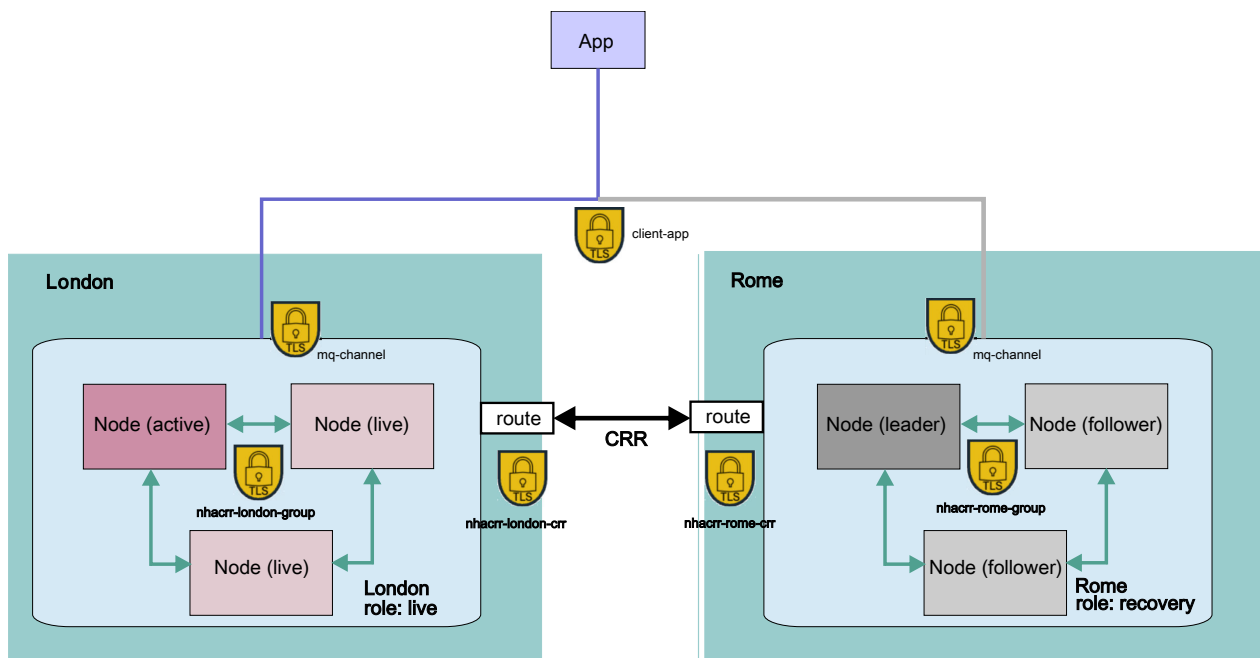
This example uses the default installation location (`/opt/mqm`) and data directory (`/var/mqm`). If you are using non-default locations, be sure to replace these paths accordingly throughout the example.

Before you begin

Before starting this example, you must first complete the exercise in [“Example: Deploying a simple Native HA CRR configuration on Linux”](#) on page 511, as the steps in this exercise assume that the deployment already exists and the London group is the live group.

Configuring TLS for Native HA CRR

The following diagram shows where in the Native HA CRR configuration certificates are used.



There are two types of certificate:

- Certificates that secure communication between the queue manager and client applications: `client-app` and `mq-channel`.
- Certificates that secure queue manager replication traffic: `nhacrr-london-group`, `nhacrr-rome-group`, `nhacrr-london-crr` and `nhacrr-rome-crr`.

Secure communication between the queue manager and the applications

The six hosts and two Native HA groups, shown in the diagram, form a highly available queue manager that client applications can connect to via the live group. When a switchover occurs, applications are disconnected and must reconnect to the new live group. You can add the connection details of all the instances in both groups to an applications connection list to enable this reconnection.

The following steps focus on the certificates for queue manager replication traffic and do not change the `mq-channel` or `client-app` certificates. The example explains how to secure the instance traffic within the Live and Recovery groups and between groups with different certificates and cipher specs. For information on securing application and queue manager communication, see [Connecting a client to a queue manager securely](#).

Secure queue manager replication traffic

Traffic is replicated both within groups and between them. The network traffic between the two Native HA CRR groups must always be secured, but the traffic within a group can be plain text.

In the diagram:

- `nhacrr-london-group` and `nhacrr-rome-group` represent the certificates used to secure traffic within each group.
- `nhacrr-london-crr` and `nhacrr-rome-crr` represent the certificates used to secure each end of the cross group traffic.

In “[Example: Deploying a simple Native HA CRR configuration on Linux](#)” on page 511, the same `nha-qm-replication` certificate is added to the `/var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb` key store and used to secure the `nhacrr-london-group`, `nhacrr-rome-group`, `nhacrr-london-crr`, and `nhacrr-rome-crr` traffic. This example shows how to use different certificates for each type of traffic.

Procedure

1. Create two certificates to use for London and Rome instance traffic:

- a. On the alpha instance in the London group run the following command to create a `nhacrr-london-group` self-signed certificate:

```
runmqakm -cert -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label nhacrr-london-group -dn CN=MYQMGR-LONDON-GRP -size 2048
```

- b. On the delta instance in the Rome group, run the following command to create a `nhacrr-rome-group` self-signed certificate:

```
runmqakm -cert -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label nhacrr-rome-group -dn CN=MYQMGR-ROME-GRP -size 2048
```

- c. To check the new certificates have been created in the key store, run the following command:

```
runmqakm -cert -list -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD
```

For a more detailed explanation of these commands, see [runmqakm and runmqtool commands on AIX, Linux, and Windows](#).

Important: The examples described here are not suitable for a production environment, and are solely intended as examples to get going quickly. Certificate management is a complex subject for advanced users. For production, you must consider things like rotation, revocation, key length, disaster recovery and much more.

2. Copy the updated key store to the instances where it will be used.

- a. Copy the four keystore files on the alpha host to the same location (`/var/mqm/qmgrs/MYQMGR/ssl/`) on the beta and gamma hosts, for example, by using `sftp`.
- b. Copy the four keystore files on the delta host to the same location (`/var/mqm/qmgrs/MYQMGR/ssl/`) on the epsilon and zeta hosts, for example, by using `sftp`.

3. Secure traffic in the London group with the new certificate. The deployment created in the [“Example: Deploying a simple Native HA CRR configuration on Linux”](#) on page 511 secured all replication traffic with the nha-qm-replication certificate. This step changes the configuration to secure the traffic within the London group with the nhacrr-london-group certificate. The group to group traffic is still secured by the nha-qm-replication certificate.

- a. On the alpha, beta, and gamma instances update the /var/mqm/qmgrs/MYQMGR/qm.ini file, change the CertificateLabel to nhacrr-london-group and add a GroupCertificateLabel:

```
CertificateLabel=nhacrr-london-group
GroupCertificateLabel=nha-qm-replication
```

- b. Restart the queue manager instances on alpha, beta, and gamma hosts so the new configuration is deployed:

```
sudo systemctl restart mqmonitor@MYQMGR
```

- c. Issue the **dspmq -o nativeha -g** command on the alpha, beta, and gamma instances and wait for the groups to have QUORUM(3/3) and GRSTATUS(Normal):

```
QMNAME(MYQMGR)                                ROLE(Active) INSTANCE(<instance-
name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME(london) GRPROLE(Live)
GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) GRSTATUS(Normal)
RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>) INITTIME(<timestamp>)
LIVETIME(<timestamp>) ALTDAT(<date>) ALTTIME(<time>)
GRPNAME(rome) GRPROLE(Recovery) GRPADDR(<leader-fqdn>) GRPVER(10.0.0.0) CONNGRP(yes)
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) BACKLOG(0) INSYNC(yes)
SYNCTIME(<timestamp>) ALTDAT(<date>) ALTTIME(<time>)
```

- d. On the active instance, check /var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG for the following messages to confirm the connections between the instances are secured with the new certificates:

```
AMQ3305I: The Native HA secure connection between the local instance and 'beta'
is using certificate DN 'CN=MYQMGR-LONDON-GRP'.
AMQ3305I: The Native HA secure connection between the local instance and 'gamma'
is using certificate DN 'CN=MYQMGR-LONDON-GRP'.
```

Look for these messages to confirm the connection between Rome and London is still secure with the nha-qm-replication certificate:

```
AMQ3308I: The Native HA secure connection between the local group and 'rome'
accepted remote certificate DN 'CN=MYQMGR-REPLICATION'.
AMQ3299I: The Native HA group connection is ready to send log data to instance
'<leader-instance>' in group 'rome'.
```

4. Secure traffic in the Rome group with the new certificate. This step changes the configuration to secure the traffic in the Rome group with the nhacrr-rome-group certificate. Again, it leaves the group to group traffic secured by the nha-qm-replication certificate.

- a. On the delta, epsilon, and zeta instances update the /var/mqm/qmgrs/MYQMGR/qm.ini file, change the CertificateLabel to nhacrr-rome-group and add a GroupCertificateLabel:

```
CertificateLabel=nhacrr-rome-group
GroupCertificateLabel=nha-qm-replication
```

- b. Restart the queue manager instances on delta, epsilon, and zeta hosts so that the new configuration is deployed:

```
sudo systemctl restart mqmonitor@MYQMGR
```

- c. Issue the **dspmq -o nativeha -g** command on the delta, epsilon, and zeta instances and wait for the groups to have QUORUM(3/3) and GRSTATUS(Normal):

```
QMNAME(MYQMGR)                                ROLE(Leader) INSTANCE(<instance-  
name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME(rome) GRPROLE(Recovery)  
GRPNAME(rome) GRPROLE(Recovery) GRPADDR(<leader-fqdn>) GRPVER(10.0.0.0)  
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) BACKLOG(0) INSYNC(yes)  
SYNCTIME(<timestamp>) ALTDATETIME(<date>) ALTTIME(<time>)  
GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) CONNGRP(yes)  
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>)  
INITTIME(<timestamp>) LIVETIME(<timestamp>) ALTDATETIME(<date>) ALTTIME(<time>)
```

- d. On the leader instance, check /var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG for the following messages to confirm the connections between the instances are secured with the new certificates:

```
AMQ3305I: The Native HA secure connection between the local instance and 'epsilon'  
is using certificate DN 'CN=MYQMGR-ROME-GRP'.  
AMQ3305I: The Native HA secure connection between the local instance and 'zeta'  
is using certificate DN 'CN=MYQMGR-ROME-GRP'.
```

Look for these messages to confirm the connection between Rome and London is still secure with the nha-qm-replication certificate:

```
AMQ3307I: The Native HA secure connection between the local group and 'london'  
is using certificate DN 'CN=MYQMGR-REPLICATION'.  
AMQ3280I: The Native HA group connection is ready to receive log data from  
instance '<active-instance>' in group 'london'.
```

5. Create a certificate for the London group to secure group to group traffic. If needed, separate certificates can be used for the within group and cross group traffic. In this step, the nhacrr-london-crr certificate used to secure the traffic between groups in London is created.

- a. On the alpha instance run the following command to create a nhacrr-london-crr self-signed certificate:

```
runmqakm -cert -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label  
nhacrr-london-crr -dn CN=MYQMGR-LONDON-CRR -size 2048
```

- b. Extract the public part of the certificate:

```
runmqakm -cert -extract -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label  
nhacrr-london-crr -target /var/mqm/qmgrs/MYQMGR/ssl/nhacrr-london-crr.crt
```

6. Trust the London certificate in Rome.

- a. Copy the nhacrr-london-crr.crt file on the alpha host to the same location (/var/mqm/qmgrs/MYQMGR/ssl/) on the delta host, for example, by using sftp.

- b. On delta, add the public part of the nhacrr-london-crr certificate to the local keystore:

```
runmqakm -cert -add -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label  
nhacrr-london-crr -file /var/mqm/qmgrs/MYQMGR/ssl/nhacrr-london-crr.crt
```

7. Create a certificate for the Rome group to secure group to group traffic. In this step, the nhacrr-rome-crr certificate used to secure the traffic between groups in Rome is created.

- a. On the delta instance in the Rome group, run the following command to create a nhacrr-rome-crr self-signed certificate:

```
runmqakm -cert -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label  
nhacrr-rome-crr -dn CN=MYQMGR-ROME-CRR -size 2048
```

- b. Extract the public part of the certificate:

```
runmqakm -cert -extract -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label  
nhacrr-rome-crr -target /var/mqm/qmgrs/MYQMGR/ssl/nhacrr-rome-crr.crt
```

8. Trust the Rome certificate in London.

- a. Copy the `nhacrr-rome-crr.crt` file on the delta host to the same location (`/var/mqm/qmgrs/MYQMGR/ssl/`) on the alpha host, for example, by using `sftp`.
- b. On alpha, add the public part of `nhacrr-rome-crr` certificate to the local keystore:

```
runmqakm -cert -add -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label
nhacrr-rome-crr -file /var/mqm/qmgrs/MYQMGR/ssl/nhacrr-rome-crr.crt
```

9. To check the new certificates are in the key store, run following command on both alpha and delta:

```
runmqakm -cert -list -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD
```

10. Copy the updated key stores to the group instances.

- a. Copy the four keystore files on the alpha host to the same location (`/var/mqm/qmgrs/MYQMGR/ssl/`) on the beta and gamma hosts, for example, by using `sftp`.
- b. Copy the four keystore files on the delta host to the same location (`/var/mqm/qmgrs/MYQMGR/ssl/`) on the epsilon and zeta hosts, for example, by using `sftp`.

11. Secure the traffic between groups with separate certificates, while leaving the `nhacrr-rome-group` and `nhacrr-london-group` certificates securing the traffic within each group. This step updates the certificates for group to group traffic. After the step is complete, the configuration matches the one shown in the diagram.

- a. On the alpha, beta, and gamma instances, update the `/var/mqm/qmgrs/MYQMGR/qm.ini` file, change the `GroupCertificateLabel` to `nhacrr-london-crr`:

```
GroupCertificateLabel=nhacrr-london-crr
```

- b. On the delta, epsilon, and zeta instances, update the `/var/mqm/qmgrs/MYQMGR/qm.ini` file. Change the `GroupCertificateLabel` to `nhacrr-rome-crr`:

```
GroupCertificateLabel=nhacrr-rome-crr
```

- c. Restart the instances in the both groups. Restart the queue manager instances on alpha, beta, gamma, delta, epsilon, and zeta hosts so the new configuration is deployed:

```
sudo systemctl restart mqmonitor@MYQMGR
```

12. Verify that the new certificates deployed correctly.

- a. Issue the **`dspmq -o nativeha -g`** command on the alpha, beta, and gamma instances and wait for the groups to have `QUORUM(3/3)` and `GRSTATUS(Normal)`. This will be the output on the active:

```
QMNAME(MYQMGR)                                ROLE(Active) INSTANCE(<instance-
name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME(london) GRPROLE(Live)
GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) GRSTATUS(Normal)
RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>) INITTIME(<timestamp>)
LIVETIME(<timestamp>) ALTDATe(<date>) ALTTIME(<time>)
GRPNAME(rome) GRPROLE(Recovery) GRPADDR(<leader-fqdn>) GRPVER(10.0.0.0) CONNGRP(yes)
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) BACKLOG(0) INSYNC(yes)
SYNCTIME(<timestamp>) ALTDATe(<date>) ALTTIME(<time>)
```

- b. Issue the **`dspmq -o nativeha -g`** command on the delta, epsilon, and zeta instances and wait for the groups to have `QUORUM(3/3)` and `GRSTATUS(Normal)`. This will be the output on the leader:

```
QMNAME(MYQMGR)                                ROLE(Leader) INSTANCE(<instance-
name>) INSYNC(yes) QUORUM(3/3) GRPLSN(<n:n:n:n>) GRPNAME(rome) GRPROLE(Recovery)
GRPNAME(rome) GRPROLE(Recovery) GRPADDR(<leader-fqdn>) GRPVER(10.0.0.0)
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) BACKLOG(0) INSYNC(yes)
SYNCTIME(<timestamp>) ALTDATe(<date>) ALTTIME(<time>)
GRPNAME(london) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) CONNGRP(yes)
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>)
INITTIME(<timestamp>) LIVETIME(<timestamp>) ALTDATe(<date>) ALTTIME(<time>)
```

- c. On the active instance, check `/var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG` for the following messages to confirm the connections between the groups in London are secured with the new certificates:

```
AMQ3307I: The Native HA secure connection between the local group and 'rome' is
using certificate DN 'CN=MYQMGR-LONDON-CRR'.
AMQ3308I: The Native HA secure connection between the local group and 'rome'
accepted remote certificate DN 'CN=MYQMGR-ROME-CRR'.
AMQ3299I: The Native HA group connection is ready to send log data to instance
'<instance-name>' in group 'rome'.
```

- d. On the leader instance, check `/var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG` for the following messages to confirm the connections between the groups in Rome are secured with the new certificates:

```
AMQ3307I: The Native HA secure connection between the local group and 'london'
is using certificate DN 'CN=MYQMGR-ROME-CRR'.
AMQ3308I: The Native HA secure connection between the local group and 'london'
accepted remote certificate DN 'CN=MYQMGR-LONDON-CRR'.
AMQ3280I: The Native HA group connection is ready to receive log data from
instance 'instance-name' in group 'london'.
```

13. Add a specific CipherSpec for Group to Group traffic.

If no specific CipherSpecs are defined for the replicated traffic then `ANY_TLS13_OR_HIGHER` is used, so that the protocols and encryption are negotiated between the instances. For further information about the CipherSpecs that IBM MQ supports, see [Enabling CipherSpecs](#).

The “[Example: Deploying a simple Native HA CRR configuration on Linux](#)” on page 511 deployment sets the CipherSpec to `ANY_TLS12` for both the traffic within and across groups. To set a different for group to group traffic, add a `GroupCipherSpec` to the `/var/mqm/qmgrs/MYQMGR/qm.ini` file on all instances:

```
GroupCipherSpec=TLS_AES_256_GCM_SHA384
```

Restart the queue manager instances on alpha, beta, gamma, delta, epsilon, and zeta hosts so the new configuration is deployed:

```
sudo systemctl restart mqmonitor@MYQMGR
```

14. Verify the new CipherSpec deployed correctly. Wait for both groups to have `QUORUM(3/3)` and `GRSTATUS(Normal)` then, on the active instance, check `/var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG` for the following message to confirm that the connections between the groups are using the `TLS_AES_256_GCM_SHA384` CipherSpec:

```
AMQ3253I: A Native HA recovery group connection to 'rome' has been established
using the 'TLS_AES_256_GCM_SHA384' CipherSpec.
```

Next on the leader instance, check `/var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG` for the following message:

```
AMQ3255I: A Native HA group connection has been accepted from group 'london'
using the 'TLS_AES_256_GCM_SHA384' CipherSpec.
```

Linux Native HA IRR

A Native HA In-Region Replication (IRR) configuration enables you to switch the running of a queue manager to a different Native HA configuration in a different location within the same region.

Note: Native HA In-Region Replication (IRR) is not currently supported in containers.

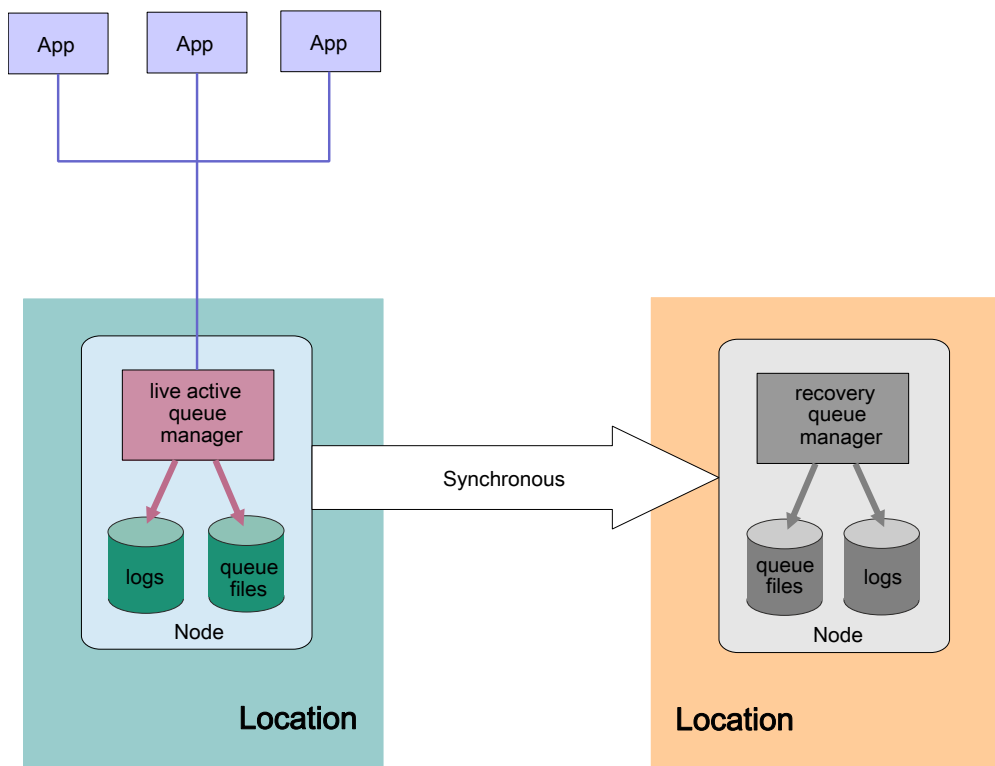
In-region replication implies that the two configurations are in two data centers that are in the same general geographical area (that is, within 100 miles or 160 kilometres of each other, with a network latency of up to 5 ms). This means that replication of data between Live and Recovery groups can be synchronous. When synchronous replication is specified, log data is streamed to the Recovery group and an acknowledgement sent in response when the data is written to disk.

Unlike Native HA or Native HA CRR, Native HA IRR has two groups each containing one instance of a queue manager.

There is no automatic failover in a Native HA IRR configuration, you must manually switch to the Recovery group, so it is intended as a disaster recovery solution (not a high availability solution). If some disaster occurs at your Live site, you can perform a failover and continue running the queue manager at the recovery site. Because replication is synchronous, the recovery logs are synchronized when the failover occurs, so there should be no loss of data. You can configure the system so that, during operation, you can prioritize either availability of a Live queue manager or having a complete set of data (RPO of zero). See [“Synchronization consistency”](#) on page 526. You can also use a Native HA IRR configuration to enable you to provide continuity while performing maintenance or testing. (See [“Complete a planned switchover on a Native HA IRR configuration”](#) on page 528 and [“Complete an unplanned failover on a Native HA IRR configuration”](#) on page 530.)

To configure a Native HA IRR solution on Linux, a second node using the same architecture is required to act as the recovery group. The resources of this node must be allocated so it can perform in either a Live or Recovery role, if required.

TLS encryption is mandatory when replicating log data between the two Native HA IRR groups.



Roles

A Requested role is the GroupRole configuration setting for a Native HA instance that is specified in the NativeHALocalInstance stanza of its `qm.ini`. The setting requests that the instance fulfills a role. See [NativeHALocalInstance stanza of the `qm.ini` file](#).

Live role

A Live group is a Native HA IRR queue manager consisting of a single instance. A Live group is also known as the active queue manager, this instance can accept application and tooling connections to perform new messaging work and manage queue manager objects.

Recovery role

A Recovery group also consists of a single instance. The instance acts only as a proxy, accepting work sent from a paired Live group. No application connections are permitted to the Recovery group,

although some control commands are allowed so that operations such as ending the queue manager instance can be completed.

The NativeHARecoverGroup stanza in the `qm.ini` configuration file of the Live group identifies the network address for the Recovery group. See [“NativeHARecoverGroup stanza of the qm.ini file” on page 150](#).

Pending role transitions

When performing a planned switchover where the current Live group becomes the new Recovery group (and the current Recovery group becomes the new Live group) there needs to be co-ordination between the groups. This co-ordination ensures that the logs are identical and achieves an RPO (recovery point objective) of zero.

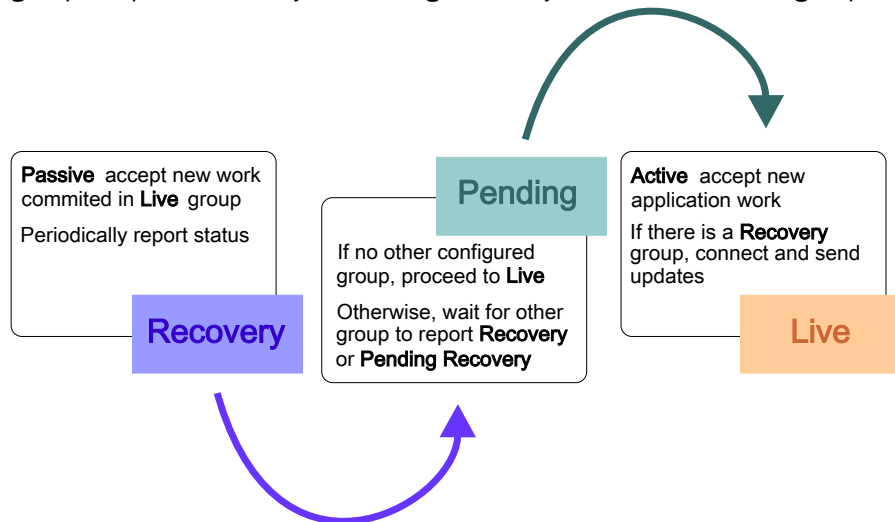
Where two groups are coordinating the switch between Recovery and Live, an effective Pending role is temporarily adopted by both groups.

Recovery role to Live role (Pending Live)

A group that has previously operated in a Recovery role that intends on changing to a Live role makes the decision on when to use the role based on whether there is another group configured.

If there is no other configured group, it transitions to the Live role immediately.

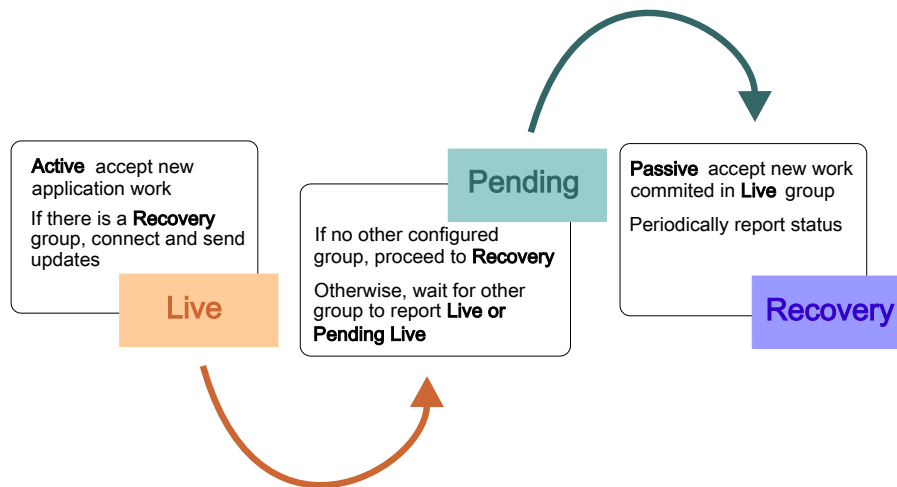
If another group is configured and enabled, the transition to a Live role is blocked until the other group adopts a Recovery or Pending Recovery role, and until both groups are in-sync.



Live role to Recovery role (Pending Recovery)

A group that had previously operated in a Live role that intends on changing to a Recovery role will make the decision on when to use the Recovery role, based on whether there is another group configured.

If another group is configured and enabled, the transition to a Recovery role is blocked until the other group adopts a Live or Pending Live role and until both groups are in-sync.



Replication and rebasing

While the two groups in a Native HA IRR configuration are connected, log updates are passed to the Recovery group. This is known as **replication**.

There might be circumstances where the logs on the Live and Recovery groups get out of sync. For example, you might be commissioning a new, currently blank, recovery group, or you might have lost the original Live group, and decided to run the queue manager and progress work on what had been the Recovery group.

In the case of out-of-sync logs, the current Recovery group performs a **rebase**. A rebase is the process of discarding the received log data and rebuilding the queue manager from a complete set of log data sent from the Live group.

If a problem is encountered during a rebase operation, then it is possible that the queue manager could be unable to start in the Recovery group. To avoid this possibility, a backup of the log is taken before the rebase operation commences. The queue manager can then be restarted using the backup log data if the rebase fails. The backup log is deleted after the rebase has succeeded.

Because of the requirement for a backup log, a Native HA IRR configuration requires at least twice the storage space of a Native HA configuration.

Instance and group naming

When configuring Native HA IRR, each instance also needs a group name. The group name is used for identification purposes (in **dspm** output and messages), and so you might want to use a geographical name to identify the group, for example 'Northampton' and 'Leicester'.

Avoid using a group name that implies the role of the group, for example 'Live' and 'Recovery' or 'primary' and 'backup', such names can be confusing in **dspm** output and messages when the 'usual' roles have been swapped between the groups.

The rules for naming groups are the same as those for naming instances. The one instance in a group forms a single queue manager, but still requires a name for identification purposes. When deploying Native HA IRR in virtual machines or on bare-metal servers you can use the hostname as the instance name to identify the hosting location (but any value would suffice, for example, d_Artagnan).

Instance names can be up to 48 characters in length. The names are case-sensitive and can include following valid characters:

- Uppercase (A-Z)
- Lowercase (a-z)
- Numerics 0-9
- Period (.)

- Forward slash (/)
- Dash (-)
- Underscore (_)
- Percent sign (%)

Synchronization consistency

Provided you have set up a Native HA IRR configuration with synchronous replication (SyncReplication=Yes in the NativeHARecoverGroup stanza of the `qm.ini` file), you can further specify whether you want to prioritize data consistency or service availability for the queue manager. By default, data consistency is prioritized and both Live and Recovery groups must be connected and in-sync before a queue manager can start and progress new log writes. The Live group active instance abdicates if it is unable to reconnect to the Recovery group within 60 seconds. The abdication forcibly disconnects applications. The default behavior is the equivalent of setting SyncConsistency to Strict in the NativeHARecoverGroup stanza of the `qm.ini` file.

You can, alternatively, set SyncConsistency to Eventual to prioritize availability. With this setting, the Live group makes a single attempt to connect to the Recovery group and synchronize when it starts. If the Live group is unable to establish a replication link, it starts to progress log writes without waiting for the acknowledgement from the Recovery group. When the Live and Recovery group are connected and both are back in-sync, the Live group progresses new log writes when both groups acknowledge the write to disk (as for the Strict consistency setting). If replication fails between Live and Recovery, you are informed by messages in the queue manager error log. You can also use the **dspmq** (or **DISPLAY QMSTATUS**) command to check that your Recovery group is not falling too far behind your Live group.

You might want to set your Recovery group to Eventual so that, in the event of you Live group failing, you can switch over to the Recovery group and start the queue manager instance there and progress work while you take steps to recover your original Live group.

Creating a Native HA IRR configuration

You need two nodes to create a Native HA In-region replication (IRR) queue manager with two instances on Linux. The nodes can be virtual machines or bare-metal servers, although you must use the same type of node for both instances.

About this task

Each node must initially be able to communicate with the other node and start with the same IBM MQ maintenance level.

You create a Native HA instance on each node by using the **crtmqm** command (although the two instances constitute a single queue manager).

In this example, the nodes have the host names alpha and beta. For simplicity, these names are also used as the instance names inside the group. The group names are datacenter1 and datacenter2.

The queue manager is called MYQMGR, and is created with ten primary and ten secondary extents of 32 MB. You can specify your own size for the logs, but you must use consistent values for first and second instances. For more information on log size options, see “Calculating the size of the log” on page 733. The `-p` option is specified to tell the queue manager to start a TCP/IP listener for channels and clients on port 1414, and start this when an instance is started as active.

After you create the two instances, you edit the `qm.ini` file for each instance to add information that defines this as an In-region replication configuration and enables them to communicate with one another. You update the NativeHALocalInstance stanza to define the local node and TLS information for that node, see NativeHALocalInstance stanza of the `qm.ini` file. You update the NativeHARecoverGroup stanza to identify the other instance in the configuration and specify the synchronous replication options, see “NativeHARecoverGroup stanza of the `qm.ini` file” on page 150. (You do not add NativeHAInstance stanza when defining a Native HA IRR configuration.)

You can view the status of a Native HA IRR queue manager by using the following methods:

- The **dspmq** command (see [dspmq \(display queue managers\)](#)).
- The IBM MQ Console (see [IBM MQ Console: Viewing queue manager status](#)).
- The IBM MQ Explorer (see [Native HA queue manager status attributes](#)).

The instructions use an example set up that has the following values:

- Queue manager:
 - Name - MYQMGR
 - Instance names - alpha and beta
 - Port for listener - 1414
- Live group:
 - Name - datacenter1
 - Node name - alpha
 - Address - alpha.example.ibm.com
 - Port for replication traffic - 4454
- Recovery group:
 - Name - datacenter2
 - Node name - beta
 - Address - beta.example.ibm.com
 - Port for replication traffic - 4454

You should substitute your own values when configuring your system.

If you have several queue managers in your Native HA IRR configurations, each would need a different replication port (although the two instances of the queue manager would use the same port). For example, the instances of MYQMGR might use 4454, while the instances of MYOTHERQMGR use 4453.

For a step-by-step example of creating a Native HA IRR configuration, including creating certificates, see [“Example: Deploying a simple Native HA IRR configuration on Linux” on page 536](#).

Procedure

1. Create the first instance on the node named alpha:

```
crtmqm -lr alpha -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

The queue manager is created:

```
IBM MQ queue manager 'MYQMGR' created.  
Directory '/var/mqm/qmgrs/MYQMGR' created.  
The queue manager is associated with installation 'Installation1'.  
Creating or replacing default objects for queue manager 'MYQMGR'.  
Default objects statistics : 85 created. 0 replaced. 0 failed.  
Completing setup.  
Setup completed.
```

2. Create the second instance on the node named beta:

```
crtmqm -lr beta -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

3. Add information to the `qm.ini` file for your instance on the alpha node, which will be your Live group. This includes setting the replication type used to communicate with the Recovery group to synchronous:

```
NativeHALocalInstance:  
  Name=alpha  
  GroupName=datacenter1  
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore
```

```
CertificateLabel=MyCertificate
GroupRole=Live
GroupCipherSpec=ANY_TLS12
GroupLocalAddress=(4454)

NativeHARecoverGroup:
  GroupName=datacenter2
  ReplicationAddress=beta.example.com(4454)
  SyncReplication=Yes
```

4. Add information to the `qm.ini` file for your instance on the beta node, which will be your Recovery group. This includes setting the replication type used to communicate with the Recovery group (when `datacenter1` occupies that role) to synchronous:

```
NativeHALocalInstance:
  Name=beta
  GroupName=datacenter2
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore
  CertificateLabel=MyCertificate
  GroupRole=Recovery
  GroupCipherSpec=ANY_TLS12
  GroupLocalAddress=(4454)

NativeHARecoverGroup:
  GroupName=datacenter1
  ReplicationAddress=alpha.example.com(4454)
  SyncReplication=Yes
```

5. You can now start the queue manager on the alpha node:

```
strmqm MYQMGR
```

You receive the following response:

```
IBM MQ queue manager 'MYQMGR' starting.
The queue manager is associated with installation 'Installation1'.
Queue manager startup is waiting to synchronize log data with the 'datacenter2' recovery
group.
Queue manager startup is waiting to synchronize log data with the 'datacenter2' recovery
group.
Queue manager startup is waiting to synchronize log data with the 'datacenter2' recovery
group.
Queue manager startup is waiting to synchronize log data with the 'datacenter2' recovery
group.
```

The waiting message continues until the Recovery group instance is started.

6. Then start the queue manager on the beta node:

```
strmqm MYQMGR
```

Wait for both **strmqm** commands to complete.

Complete a planned switchover on a Native HA IRR configuration

You can perform a planned switchover between the Live and Recovery groups in a Native HA In-Region Replication (IRR) configuration.

About this task

You might require to switch from your Live group to your Recovery group if you want to test your configuration, or need to apply maintenance. (If you upgrading IBM MQ, see [“Upgrading Native HA configurations”](#) on page 496 for guidance.)

When you perform a planned switchover, the system takes steps to ensure that you do not lose data or encounter a partitioned (split-brain) problem. You request the switchover by changing the group role on each of the groups. For a while both groups enter a pending status while their Recovery logs are synchronized. When this is achieved, the groups adopt their new roles and the new Live group and takes the active role and runs the queue manager.

This task uses the example configuration created in [“Creating a Native HA IRR configuration”](#) on page 526.

Procedure

1. Specify that the Recovery group, datacenter2, should become the Live group by editing the qm.ini file of the queue manager instance. Change the GroupRole to Live, and ensure NativeHARecoveryGroup Enabled is Yes.

```
NativeHALocalInstance:
  . . .
  GroupName=datacenter2
  GroupRole=Live

NativeHARecoveryGroup:
  GroupName=datacenter1
  . . .
  Enabled=Yes
```

2. Restart the instance in the datacenter2 group:

```
endmqm MYQMGR
strmqm MYQMGR
```

After the restart, the instance moves to the pending Live role, followed by the Live role when the instances are in sync. The **dspmq -m MYQMGR -o nativeha -g** command confirms the instance role is PendingLive.

Note: While the instance is in the pending Live role, you can change your mind about the switchover and revert to the previous Live and Recovery arrangement. Reverse the changes that you made in steps 1 and 3 and restart the instances.

3. Specify that the Live group, datacenter1, should become the Recovery group by editing the qm.ini file for the queue manager instance. Change the GroupRole to Recovery, and ensure NativeHARecoveryGroup Enabled is Yes.

```
NativeHALocalInstance:
  . . .
  GroupName=datacenter1
  GroupRole=Recovery

NativeHARecoveryGroup:
  GroupName=datacenter2
  . . .
  Enabled=Yes
```

Changing a group role from Live to Recovery, or Recovery to Live with this configuration instructs each group to co-ordinate its role transition with the other. While the negotiation is taking place, each group adopts a pending role which indicates the role it is in the process of changing to.

4. Restart the instance in the datacenter1 group:

```
endmqm MYQMGR
strmqm MYQMGR
```

After the restart, the **dspmq -m MYQMGR -o nativeha -g** command should confirm the pending Recovery role.

What to do next

When the maintenance is complete, you can switch back to the original Live group by reversing the process, restoring datacenter1 as the Live group and datacenter2 as the Recovery group.

Complete an unplanned failover on a Native HA IRR configuration

You perform an unplanned failover between the Live and Recovery groups in a Native HA In-Region Replication (IRR) configuration when some disaster occurs at the Live site and it can no longer operate.

About this task

This task uses the example configuration created in [“Creating a Native HA IRR configuration”](#) on page 526.

When the Live group goes down, the Recovery group cannot detect this as it simply stops receiving replicated log data, which could just mean there is no ongoing workload. An unplanned failover must therefore be manually carried out by changing the Recovery group's role to Live and notifying it that the previously Live group is disabled so that it does not attempt to replicate data to it.

The Recovery group should be in-sync with the Live group as log data is replicated synchronously. If you have accepted the default setting of prioritizing data (`SyncConsistency=Strict`), replication from `datacenter1` stops if it loses connection to `datacenter2`. If you have set `SyncConsistency=Eventual` to prioritize service availability, however, it is possible that the Recovery group is not in-sync with the Live group and data might be lost when an unplanned failover occurs. Before initiating an unplanned failover, issue the `dspsmq -o nativeha -g` command on the Recovery group, and use the `RCOVLSN` and `RCOVTIME` values to view the log sequence number (LSN) and time that the group can recover to. For guidance, see step “6” on page 531 and [“Switchover and Recovery points”](#) on page 535.

Important: If you have partitioned (split-brain) data, you must follow these steps carefully to avoid unwanted permanent data loss.

In this example, the `datacenter1` groups fails, and the `datacenter2` group assumes the Live role.

Procedure

1. Ensure that applications have disconnected from the `datacenter1` group, and are configured to reconnect to the `datacenter2` group (which is currently the Recovery group).
2. Stop the queue manager instance in the `datacenter1` group (to mimic losing the node in `datacenter1`):

```
endmqm MYQMGR
```

3. Modify the `/var/mqm/qmgrs/MYQMGR/qm.ini` file for the instance in the `datacenter2` group, changing the group role from Recovery to Live:

```
NativeHALocalInstance:
.
.
.
  GroupName=datacenter2
  GroupRole=Live
.
.
.
NativeHARecoveryGroup:
.
.
.
  Enabled=No
.
.
.
```

Because the `NativeHARecoveryGroup` stanza in the `datacenter2` group configuration is disabled (`Enabled` is set to `No`), the group processes this change as an unplanned failover and does not attempt to co-ordinate the change with the `datacenter1` group.

4. Restart the instance in the `datacenter2` group:

```
endmqm MYQMGR
strmqm MYQMGR
```

If you issue a `dspsmq -o nativeha -g` command for the queue manager `MYQMGR` after the instance has restarted, you will see a `GRPROLE` of `Live`.

5. To restart replication after the failover, you can either use the original Live group as the Recovery group, or create a new Recovery group at a new site.

- If the original Live site is restored, before restarting the original Live group, change the role of the group to recovery. This ensures that the group does not resume its Live role.
- If the original Live site cannot be restored, create a new Native HA Recovery group at a new site.

Before choosing either option, see [“Swapping Live and Recovery roles after an unplanned failure”](#) on page 532 for guidance.

6. If the configuration on the original Live site is not changed, and the group resumes its Live role when it is restored, you will have two active Live instances of the queue manager. If this is the case, the partitioned state is detected and no more replication occurs between the two groups. You must resolve which data to keep before you restore the original Live and Recovery roles. Take the following steps to resolve a partitioned (split-brain) situation:

- a) Confirm that you have a partitioned state by issuing the following command in both groups:

```
dspmqr -o nativeha -g
```

the status GRSTATUS(Partitioned) is returned

- b) Decide which of the two sets of log data that you want to keep, and which you want to discard, by completing the following steps.
- c) Stop both instances by using the **endmqm** command. Both queue managers must be stopped to allow the **dmpmqlog** command to run in the next step.
- d) Use the following command to examine the contents of a log from the LSN identified in step b:

```
dmpmqlog -s startLSN
```

Use **dspmqr** and **dmpmqlog** in combination to identify log records that were written by one group and not replicated to the other (see **dspmqr** and **dmpmqlog**). The **dmpmqlog** command also reports a summary of the messaging and transaction activity, which can be useful when making the choice.

- e) For the group whose data you are discarding and is to become the Recovery group (for example, datacenter1 from the earlier failover scenario), edit the `/var/mqm/qmgrs/MYQMGR/qm.ini` file, set `GroupRole=Recovery` in the `NativeHALocalInstance` stanza and ensure `Enabled=No` is set in the `NativeHARecoverGroup` stanza. This action acknowledges that the unreplicated log data in this group is no longer required.



Warning: This is a point of no return. Setting `Enabled=No` acknowledges that any unreplicated log data in the recovery group will be permanently discarded and cannot be recovered. Ensure you have confirmed which data set to keep before proceeding.

- f) For the group whose data you are keeping and is to remain the Live group (for example, datacenter2), edit the `/var/mqm/qmgrs/MYQMGR/qm.ini` file and ensure `Enabled=No` is **not** set in the `NativeHARecoverGroup` stanza to allow replication to the Recovery group.
- g) Restart both the instances by using the **strmqm** command. Replication is resumed and the logs of the Recovery group are discarded and overwritten by the logs from the Live group.
- h) After replication has been re-established and is functioning correctly, remove `Enabled=No` from the `NativeHARecoverGroup` stanza in the configuration file for the Recovery group (the group that was discarded). This cleanup step ensures the configuration is returned to a clean state with no lingering values that could affect future failover operations.

Related information

[dspmqr \(display queue managers\)](#)

[dmpmqlog \(dump MQ formatted log\)](#)

Swapping Live and Recovery roles after an unplanned failure

After an unplanned failure has occurred and you have failed over to your Recovery group, you might want to reconfigure so that your original Live group is restored as the new Recovery group.

About this task

In the example configuration described in “Creating a Native HA IRR configuration” on page 526, datacenter1 is the Live group and datacenter2 is the Recovery group. If there was a major power outage at the datacenter1 site, for example, you would switch over to the datacenter2 site using the process described in “Complete an unplanned failover on a Native HA IRR configuration” on page 530. When power is restored to the datacenter1 site, you might decide to keep datacenter2 as the Live group, and make datacenter1 the Recovery group. There are some special considerations when you take this course of action.

The decision about which site will be Live and which will be Recovery after an unplanned failover must take into consideration the state of the data on each of the sites. Because the replication data in a Native HA IRR configuration is synchronized, the data should be the same at the point of failure, but it is always worth considering the state of the data on each site, because log data and queue manager data on the new Recovery site will be deleted and overwritten with data from the Live site when you reconfigure.

If you decide to swap roles so that your original Live group becomes the new Recovery group, there are two alternative procedures that you can follow. Both are completed on the original Live configuration, which is now to become the Recovery group (datacenter1 in the example):

- Delete and recreate the queue manager - data is deleted and replaced when the Live group in datacenter2 connects.
- Change the queue manager .ini file and restart - when the Live group connects a rebase occurs and data is replaced.

In both cases, you also need to change the configuration of the original Live group to identify its new Recovery role.

Procedure

- To swap roles by deleting and recreating the queue manager:
 - a) Delete the queue manager on the previous Live site in datacenter1 (in the example, the queue manager is called MYQMGR):

```
endmqm MYQMGR
dltmqm MYQMGR
```

All the log and queue manager data for this instance of MYQMGR are deleted.

- b) Recreate the queue manager, specifying that it is a Native HA IRR queue manager with replicated logging (in the example, the node name is alpha):

```
crtmqm -lr alpha -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

- c) Specify that this instance of the queue manager is in the Recovery role by editing the `qm.ini` file:

```
NativeHALocalInstance:
  Name=alpha
  GroupName=datacenter1
  KeyRepository=/var/mqm/qmgrs/MYQM/ssl/key
  CertificateLabel=MyCertificate
  CipherSpec=NULL
  GroupRole=Recovery
  GroupCipherSpec=ANY_TLS12
  GroupLocalAddress=(4453)

NativeHARecoveryGroup:
  GroupName=beta
  ReplicationAddress=beta.example.ibm.com(4454)
  SyncReplication=Yes
```


- d) Ensure that the queue manager on the new Live site (datacenter2) is correctly configured to have datacenter1 as its Recovery group:

```
NativeHALocalInstance:
  Name=beta
  GroupName=datacenter2
  KeyRepository=/var/mqm/qmgrs/MYQM/ssl/key
  CertificateLabel=MyCertificate
  CipherSpec=NULL
  GroupRole=Live
  GroupCipherSpec=ANY_TLS12
  GroupLocalAddress=(4454)

NativeHARecoveryGroup:
  GroupName=datacenter1
  ReplicationAddress=alpha.example.ibm.com(4453)
  SyncReplication=Yes
```

- e) You can now start the queue manager on datacenter1:

```
strmqm MYQMGR
```

Data is replicated to MYQMGR on datacenter1 from the Live instance on datacenter2.

- To swap roles by changing the queue manager `qm.ini` file:
 - a) Specify that the instance of the queue manager MYQMGR on datacenter1 is in the Recovery role by editing the `qm.ini` file:

```
NativeHALocalInstance:
  Name=alpha
  GroupName=datacenter1
  KeyRepository=/var/mqm/qmgrs/MYQM/ssl/key
  CertificateLabel=MyCertificate
  CipherSpec=NULL
  GroupRole=Recovery
  GroupCipherSpec=ANY_TLS12
  GroupLocalAddress=(4453)

NativeHARecoveryGroup:
  GroupName=datacenter2
  ReplicationAddress=beta.example.ibm.com(4454)
  SyncReplication=Yes
```

- b) Ensure that the queue manager on the new Live site (datacenter2) is correctly configured to have datacenter1 as its Recovery group:

```
NativeHALocalInstance:
  Name=beta
  GroupName=datacenter2
  KeyRepository=/var/mqm/qmgrs/MYQM/ssl/key
  CertificateLabel=MyCertificate
  CipherSpec=NULL
  GroupRole=Live
  GroupCipherSpec=ANY_TLS12
  GroupLocalAddress=(4454)

NativeHARecoveryGroup:
  GroupName=datacenter1
  ReplicationAddress=alpha.example.ibm.com(4453)
  SyncReplication=Yes
```

- c) You can now start the queue manager on datacenter1:

```
strmqm MYQMGR
```

The Live instance on datacenter2 performs a rebase to bring the instance on datacenter1 into sync.

Upgrading Native HA IRR configurations

How to upgrade your Native HA IRR configuration to a new version of IBM MQ.

About this task

When planning your upgrade, you should regard the queue manager instances as together constituting a single queue manager. You should not view the instances as being independent entities. Because of this, you should aim to upgrade both groups as closely together as possible.

In a Native HA IRR configuration, the instance in the group performing the Recovery role must be at an IBM MQ version equal to or higher than the group performing the Live role. You cannot fail over or switch over to a group that is a lower version. If your configuration has the default setting of `SyncConsistency=Strict`, the instance on the Live system will stop if it cannot replicate log data to the Recovery system. Because of this, you should apply upgrades to all members of the Recovery group before applying it to all members of the Live group. The upgrade of IBM MQ is only complete when the instances on both the Recovery and Live groups have been upgraded.

Note: You can update to a new fixpack (that is, the 'F' from 'V.R.M.F') without affecting the availability of queue manager instances in the described way, but it is still recommended that you upgrade all instances at the same time.

There are two approaches to upgrading your Native HA IRR configuration, depending on how much down time you can afford:

- You can upgrade your Recovery group instance, then upgrade your Live group instance. You do not perform any switch over, so the queue manager is unavailable during upgrade.
- You can upgrade your Recovery group instance, then switch over to this group while you upgrade your previous Live group. You can switch back on completion of the upgrade. Using this approach, you might have to make temporary change to your `SyncConsistency` setting. The queue manager is only unavailable while you update the `qm.ini` (by restarting the queue manager) and perform the switch overs.

Procedure

- To upgrade without performing a switch over:
 - a) On the Recovery group, stop your queue managers, for example:

```
endmqm MYQMGR
```

- b) Upgrade the node hosting your Recovery group to the new version of IBM MQ. See [Upgrading an IBM MQ installation on Linux](#).
- c) On the Live group, stop your queue managers.
- d) Upgrade the node hosting your Live group to the new version of IBM MQ.
- e) Restart your queue managers on the Recovery group. For example:

```
startmqm MYQMGR
```

- f) Restart your queue managers on the Live group.

- To upgrade with a planned switchover:
 - a) On the Live group, set `SyncConsistency=Eventual` in the `qm.ini` `NativeHARecoveryGroup` definition for this group. This ensures that queue managers can continue to run on the Live group while you upgrade the Recovery group. You must restart the queue manager to ensure that the change to `SyncConsistency` takes effect:

```
endmqm MYQMGR  
startmqm MYQMGR
```

- b) On the Recovery group, stop your queue managers, for example:

```
endmqm MYQMGR
```

- c) Upgrade the node hosting your Recovery group to the new version of IBM MQ. See [Upgrading an IBM MQ installation on Linux](#).
- d) Complete a switchover to the Recovery group that you have just upgraded. See [“Complete a planned switchover on a Native HA IRR configuration”](#) on page 528.
- e) Ensure that SyncConsistency=Eventual is set in the qm.ini NativeHARecoveryGroup definition for this group. This ensures that queue managers can continue to run on this group while you upgrade the other group.
- f) Upgrade the node hosting your original Live group to the new version of IBM MQ.
- g) Switch back to your original Live group and set SyncConsistency=Strict once more. You must restart the queue manager to ensure that the change to SyncConsistency takes effect:

```
endmqm MYQMGR  
startmq MYQMGR
```

Switchover and Recovery points

Whenever you perform a planned or unplanned switchover on a Native HA IRR configuration, it has implications for your queue manager data that you should be aware of.

Important: Both forms of switchover result in the loss of all non-persistent messages and non-durable subscriptions.

You can examine logs on Live and Recovery groups to ascertain if a switchover is likely to result in any loss of persistent messages or durable subscriptions because the logs are out of sync. You use the **dspmqr** command to see data about the most recent changes to the logs, and the **dmpmqlog** command to examine the actual contents of the logs. You must stop the queue manager instances using **endmqm** before you examine the state of the logs.

To see data about the most recent changes that you could retain after a switchover, use the following command on your Recovery instance:

```
dspmqr -o nativeha -g
```

This returns status information. For example, a Recovery instance might return the following information when the logs:

```
QMNAME(MYQMGR)                                ROLE(Leader) INSTANCE(beta)  
INSYNC(yes) QUORUM(1/1) GRPLSN(<n:n:n:n>) GRPNAME(datacenter1) GRPROLE(Recovery)  
GRPNAME(datacenter1) GRPROLE(Recovery) GRPADDR(<beta-fqdn>) GRPVER(10.0.0.0) GRSTATUS(Normal)  
RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) BACKLOG(0) INSYNC(yes) SYNCTIME(<timestamp>)  
ALTDATE(<date>) ALTTIME(<time>)  
GRPNAME(datacenter2) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) CONNGRP(yes)  
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>)  
INITTIME(<timestamp>) LIVETIME(<timestamp>) ALTDATE(<date>) ALTTIME(<time>)
```

The following fields are significant for this task:

INSYNC

If this is set to yes, the Recovery log is in sync with the Live log and you could perform a switchover with no loss of data. If the logs are not in sync, then you can use other fields to assess how much data would be lost.

BACKLOG

If the logs are not in sync, this shows how far the Recovery log is behind the Live log, in KB.

RCOVLSN

A log sequence number (LSN) that the group could recover to, in nnnnn:nnnnn:nnnnn:nnnnn format. Any data written to the Recovery log after this point might be lost if an unplanned failover occurs. This value can be passed into the -s option of the **dmpmqlog** command.

RCOVRTIME

A time that the group could recover to, in ISO 8601 format. Any data written to the Recovery log after this time might be lost if an unplanned failover occurs. This value can be passed into the -t or -u options of the **dmpmqlog** command.

You can use the **dmpmqlog** command on the Live and Recovery instances to view the actual log data that might be lost in the case of a switchover.

Related information

[dspmq \(display queue managers\)](#)

[dmpmqlog \(dump MQ formatted log\)](#)

Example: Deploying a simple Native HA IRR configuration on Linux

This example installs a queue manager using a simple Native HA In-Region Replication (IRR) configuration across two Linux hosts situated in two data centers within the same geographical region.

Before you begin

To complete this example, you must first complete the following tasks:

- Follow the steps in [Checking requirements on Linux](#).
- Follow the steps in [Planning to install IBM MQ on Linux](#)
- For each of the two Linux hosts, follow the steps in [Preparing the system on Linux](#).
- Download the 10.0.0 or above IBM MQ base install images for your Linux distribution (see [IBM MQ downloads](#)).

About this task

The Native HA IRR configuration defines two Native HA groups, which together form a single queue manager. Because the configuration uses synchronous replication, the groups must be located within 100 miles (160 km) of one another, with a network latency of 5 ms or less. For more information about Native HA IRR configurations on Linux, see “Native HA IRR” on page 522.

This example defines a Northampton group and a Leicester group, each consisting of one IBM MQ instance installed on a Linux host. The host names of the Northampton group is Northampton-instance and that of the Leicester group is Leicester-instance. Throughout the example, replace these names with the host names of your systems. For simplicity, this example uses a single keystore to secure all the queue manager instance and group traffic.

This example walks through defining two Native HA groups on Linux hosts and providing in-region replication and performing a switchover.

This example uses the default installation location (/opt/mqm) and data directory (/var/mqm). If you are using non-default locations, be sure to replace these paths accordingly throughout the example.

Procedure

1. Install IBM MQ on both Linux hosts.

Follow the instructions in [Installing and uninstalling IBM MQ on Linux](#) to accept the IBM MQ license and install the IBM MQ server on both of your hosts.

For this example, it is easiest to use the default locations for installation (/opt/mqm) and data (/var/mqm). If you change the install or data location on one system, ensure that you mirror the same change across the other host.

2. Set up environments ready to create the Native HA instances:

- a) On each Linux host, log in as the mqm user and set up your environment by using the following command:

```
. /opt/mqm/bin/setmqenv -s
```

- b) Check that the environment is set up correctly by entering the following command:

```
dspmqr
```

If the command completes successfully, and the expected version number and installation name are returned, the environment is set up correctly.

3. Create the Northampton group queue manager instance. On host Northampton-instance, create a queue manager with log replication. (You might want to use the AutoConfig stanza to ensure objects are always defined in a uniform way, the **-ic** option can be added to **crtmqm** to supply a mqsc file for the queue manager.)

- Create the Northampton queue manager instance on the node named Northampton-instance:

```
crtmqm -lr Northampton-instance -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

The queue manager is created:

```
IBM MQ queue manager 'MYQMGR' created.  
Directory '/var/mqm/qmgrs/MYQMGR' created.  
The queue manager is associated with installation 'Installation1'.  
Creating or replacing default objects for queue manager 'MYQMGR'.  
Default objects statistics : 85 created. 0 replaced. 0 failed.  
Completing setup.  
Setup completed.
```

4. Create the Leicester queue manager instance. As the mqm user, on host Leicester-instance create a queue manager with log replication. (If you used the AutoConfig stanza for the Northampton instance, use the **crtmqm -ic** option for this instances as well.)

- Create the Leicester queue manager instance on the node named Leicester-instance:

```
crtmqm -lr Leicester-instance -lf 8192 -lp 10 -ls 10 -p 1414 MYQMGR
```

5. Create a keystore for securing the group traffic.

- a) To create this keystore, log on to the Northampton-instance host and run the following commands:

```
runmqakm -keydb -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -stash  
runmqakm -cert -create -db /var/mqm/qmgrs/MYQMGR/ssl/keystore.kdb -pw PASSWORD -label  
nha-qm-replication -dn CN=MYQMGR-REPLICATION -size 2048
```

- b) Copy the four keystore files to the same location (`/var/mqm/qmgrs/MYQMGR/ssl/`) on the Leicester-instance host, for example, by using sftp.

- c) On each host (Northampton-instance and Leicester-instance), ensure that the files have mqm group ownership and have read access for the mqm group:

```
chown :mqm /var/mqm/qmgrs/MYQMGR/ssl/keystore.*  
chmod g+r /var/mqm/qmgrs/MYQMGR/ssl/keystore.*
```

Important: The examples described here are not suitable for a production environment, and are solely intended as examples to get going quickly. Certificate management is a complex subject for advanced users. For production, you must consider things like rotation, revocation, key length, disaster recovery and much more.

6. Create the Northampton group on the Northampton-instance host with the initial role of Live. Later, the Leicester-instance host will form the Leicester group, and its initial role will be Recovery. The groups are created by editing the `/var/mqm/qmgrs/MYQMGR/qm.ini` files for each queue manager instance.

- a) Update the `NativeHALocalInstance` stanza on Northampton-instance to define the group and the certificate to use to secure the instance traffic:

```
NativeHALocalInstance:
  Name=Northampton-instance
  GroupName=Northampton
  GroupRole=Live
  GroupLocalAddress=(4454)
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore
  GroupCertificateLabel=nha-qm-replication
```

Where:

- `Northampton-instance` is the name of the instance that you are configuring.
- `GroupCertificateLabel` specifies the certificate for group to group traffic.

- b) Update the `NativeHARecoveryGroup` stanza on Northampton-instance to define the Recovery group:

```
NativeHARecoveryGroup:
  GroupName=Leicester
  ReplicationAddress=Leicester-instance.example.com(4454)
  SyncReplication=Yes
```

This creates an IRR configuration using Strict SyncConsistency. For more details on SyncConsistency options, see [“NativeHARecoveryGroup stanza of the qm.ini file” on page 150](#).

If you want to use Eventual SyncConsistency instead, add `SyncConsistency=Eventual` beneath `SyncReplication=Yes`

7. Create the Leicester group on the Leicester-instance host.

- a) Edit the `/var/mqm/qmgrs/MYQMGR/qm.ini` file for the host. Add the following to the `NativeHALocalInstance` stanza:

```
NativeHALocalInstance:
  Name=Leicester-instance
  GroupName=Leicester
  GroupRole=Recovery
  GroupLocalAddress=(4454)
  KeyRepository=/var/mqm/qmgrs/MYQMGR/ssl/keystore
  GroupCertificateLabel=nha-qm-replication
```

- b) Update the `NativeHARecoveryGroup` stanza on Leicester-instance to define the Recovery group:

```
NativeHARecoveryGroup:
  GroupName=Northampton
  ReplicationAddress=Northampton-instance.example.com(4454)
  SyncReplication=Yes
```

If you have used Eventual SyncConsistency in step 6b, add `SyncConsistency=Eventual` beneath `SyncReplication=Yes`.

8. Start both instances of the queue managers:

- a) Start the queue manager instance in the Northampton group.

```
strmqm MYQMGR
```

If Strict SyncConsistency is set, you receive the following response:

```
IBM MQ queue manager 'MYQMGR' starting.
The queue manager is associated with installation 'Installation1'.
Queue manager startup is waiting to synchronize log data with the 'Leicester' recovery
group.
Queue manager startup is waiting to synchronize log data with the 'Leicester' recovery
group.
Queue manager startup is waiting to synchronize log data with the 'Leicester' recovery
group.
Queue manager startup is waiting to synchronize log data with the 'Leicester' recovery
group.
```

The waiting message continues until the Recovery group instance is started.

If Eventual SyncConsistency is used, the **strmqm** command completes.

- b) Start the queue manager instance in the Leicester group.

```
strmqm MYQMGR
```

Wait for both **strmqm** commands to complete.

- c) Verify the Leicester group has deployed correctly. Issue the **dspmq -o nativeha -g** command on Leicester-instance, to get the group status. When the groups are in-sync, the status is:

```
QMNAME(MYQMGR)                                ROLE(Leader)
INSTANCE(Leicester-instance) INSYNC(yes) QUORUM(1/1) GRPLSN(<n:n:n:n>)
GRPNAME(Leicester) GRPROLE(Recovery)
  GRPNAME(Leicester) GRPROLE(Recovery) GRPADDR(<Leicester-instance-fqdn>)
GRPVER(10.0.0.0) GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) BACKLOG(0)
INSYNC(yes) SYNCTIME(<timestamp>) ALTDATE(<date>) ALTTIME(<time>)
  GRPNAME(Northampton) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) CONNGRP(yes)
GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>)
INITTIME(<timestamp>) LIVETIME(<timestamp>) ALTDATE(<date>) ALTTIME(<time>)
```

The key parts of this status are:

- ROLE(Leader)
- GRPNAME(Leicester) GRPROLE(Recovery) - Leicester is the Recovery group
- GRPNAME(Northampton) - Northampton is the live group

- d) Verify the Northampton group has deployed correctly. Issue the **dspmq -o nativeha -g** command on the Northampton-instance host to display the group status. The status is:

```
QMNAME(MYQMGR)                                ROLE(Active)
INSTANCE(Northampton-instance) INSYNC(yes) QUORUM(1/1) GRPLSN(<n:n:n:n>)
GRPNAME(Northampton) GRPROLE(Live)
  GRPNAME(Northampton) GRPROLE(Live) GRPADDR(Unknown) GRPVER(10.0.0.0) GRSTATUS(Normal)
RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>) INITLSN(<n:n:n:n>) INITTIME(<timestamp>)
LIVETIME(<timestamp>) ALTDATE(<date>) ALTTIME(<time>)
  GRPNAME(Leicester) GRPROLE(Recovery) GRPADDR(<Leicester-instance-fqdn>)
GRPVER(10.0.0.0) CONNGRP(yes) GRSTATUS(Normal) RCOVLSN(<n:n:n:n>) RCOVTIME(<timestamp>)
BACKLOG(0) INSYNC(yes) SYNCTIME(<timestamp>) ALTDATE(<date>) ALTTIME(<time>)
```

The key parts of the status are:

- ROLE(Active)
- GRPNAME(Northampton) GRPROLE(Live) - Northampton is the live group
- GRPNAME(Leicester) - the Recovery group will be Leicester.

- e) On the Leicester host, look for these messages in the `/var/mqm/qmgrs/MYQMGR/errors/AMQERR01.LOG` to confirm a secure connection between Northampton and Leicester, and that the connection is ready to receive data:

```
AMQ3307I: The Native HA secure connection between the local group and 'Northampton'
is using certificate DN 'CN=MYQMGR-REPLICATION'.
AMQ3280I: The Native HA group connection is ready to receive log data from
instance 'Northampton-instance' in group 'Northampton'.
```

9. Define a queue. On the active instance, start MQSC by entering the following command:

```
runmqsc MYQMGR
```

A message indicates when MQSC starts. MQSC has no command prompt. Define a local persistent queue:

```
DEFINE QLOCAL('QUEUE1') DEFPSIST(YES)
```

A message indicates when the queue is created. Stop MQSC by entering the following command:

```
end
```

Messages are shown, followed by the command prompt. The QUEUE1 queue is replicated to the other instances in both the London and Rome groups.

10. Add persistent messages to the queue. The DEFPSIST (YES) attribute on the QUEUE1 queue causes messages put to the queue with the amqsputc sample to be persistent. They are replicated to the Recovery group.

Change directory into /opt/mqm/samp/bin, which contains the sample programs. Put messages on the queue by entering the following commands:

```
./amqsput QUEUE1 MYQMGR
```

If connection to the queue manager is successful, the following response is output:

```
target queue is QUEUE1
```

Put several messages to the queue, by entering some text then pressing Enter each time. Enter a blank line to end the message input. The following message is shown:

```
Sample AMQSPUT0 end
```

Your messages are now on the queue and the command prompt is shown.

11. Perform a managed switchover, following the instructions in [“Complete a planned switchover on a Native HA IRR configuration”](#) on page 528.
12. Retrieve messages sent before the switchover. On the current active instance, change directory into /opt/mqm/samp/bin, which contains the sample programs. Get the messages from the queue by entering the following command:

```
./amqsget QUEUE1 MYQMGR
```

The messages you added in step 10 are available, and are output. After a few seconds, the command exits.

What to do next

When the maintenance is complete, you can switch back to the original live group by repeating the process in steps 11.

You could also try an unplanned failover using the instructions in [“Complete an unplanned failover on a Native HA IRR configuration”](#) on page 530.

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 69 on page 541 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. (There can only be one active instance and one standby instance of the queue manager in a multi-instance configuration.)

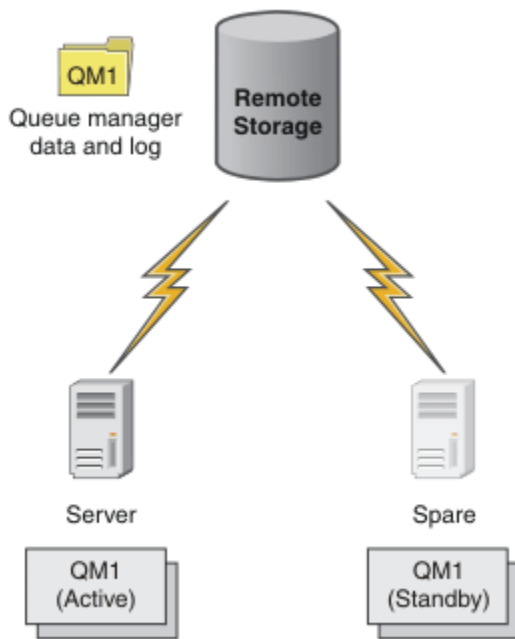


Figure 69. Multi-instance queue manager

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 Multiplatforms system.

Automatic client reconnection is supported for MQI clients.

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 AIX 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 **dspmqlinf** 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 AIX 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 [“Securing shared queue manager data and log directories and files on Windows” on page 569](#). 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, [“Creating a multi-instance queue manager on domain workstations or servers on Windows” on page 545](#), 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, [“Creating a multi-instance queue manager on Windows domain controllers” on page 559](#), 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, [“Creating a multi-instance queue manager on Linux” on page 581](#), [“Creating a multi-instance queue manager on domain workstations or servers on Windows” on page 545](#), and [“Creating a multi-instance queue manager on Windows domain controllers” on page 559](#) are examples showing how to configure a multi-instance queue manager.

On AIX, Linux, and Windows 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.  For example, the network share set up on a Linux server could contain separate queue manager data and logs for AIX and Linux queue managers, but could not contain the queue manager data used by IBM i.

 You can create multiple shares on the same networked storage for IBM i and for AIX and Linux 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 AIX and Linux, and IBM i. The fact that the user and group can have the same uid and gid does not relax the restriction.

5. On AIX 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 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.

Important: By default, computers beginning with Windows 10 version 1607 and Windows Server 2016 are more restrictive than earlier versions of Windows.

This change restricts clients allowed to make remote calls to the Security Accounts Manager (SAM) and could impact IBM MQ with queue managers failing to start. Access to SAM is critical for the functioning of IBM MQ when IBM MQ is configured as a domain account.

Before configuring a multi-instance queue manager, read [“Secure unshared queue manager data and log directories and files on Windows” on page 572](#) and [“Securing shared queue manager data and log directories and files on Windows” on page 569](#) 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 [“Creating a multi-instance queue manager on domain workstations or servers on Windows” on page 545](#).

Run a multi-instance queue manager on domain workstations or servers

From IBM WebSphere MQ 7.1, multi-instance queue managers run on a workstation or server that is a member of a domain. 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.

See [“Creating a multi-instance queue manager on domain workstations or servers on Windows” on page 545](#) for details on creating a multi-instance queue manager.

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 572](#). 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 574](#) and [“Reading and writing data and log files authorized by an alternative local security group” on page 577](#). Another topic, [“Securing shared queue manager data and log directories and files on Windows” on page 569](#), 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 on Windows” on page 548.](#)
2. [“Installing IBM MQ on a server or workstation in a Windows domain” on page 551.](#)
3. [“Creating a shared directory for queue manager data and log files on Windows” on page 554.](#)
4. [“Reading and writing shared data and log files authorized by an alternative global security group” on page 556.](#)

You can then do the task, [“Creating a multi-instance queue manager on domain workstations or servers on Windows” on page 545](#), 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

Queue manager data could be secured with the domain mqm group. As the topic [“Securing shared queue manager data and log directories and files on Windows” on page 569](#) 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, [“Creating a multi-instance queue manager on Windows domain controllers” on page 559](#) 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](#)

Windows *Creating a multi-instance queue manager on domain workstations or servers on Windows*

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 AIX 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 on Windows” on page 548.](#)
2. [“Installing IBM MQ on a server or workstation in a Windows domain” on page 551.](#)
3. [“Creating a shared directory for queue manager data and log files on Windows” on page 554.](#)
4. [“Reading and writing shared data and log files authorized by an alternative global security group” on page 556.](#)

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 551](#) 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 556](#), to create it
 - b) Check the values of the `Prefix` and `InstallationName` parameters are correct for *venus*.

On *mars*, run the **dspmqlinf** command:

```
dspmqlinf 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 **dspmqlinf** command again, with the `-o` command parameter.

```
dspmqlinf -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 [“Verifying the multi-instance queue manager on Windows” on page 566](#).

Windows *Creating an Active Directory and DNS domain on Windows*

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 544.

For the purposes of the task the domain controller host name 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\Administrator*.

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 551.

Related tasks

Windows [Installing IBM MQ on a server or workstation in a Windows domain](#)

Windows [Creating a shared directory for queue manager data and log files on Windows](#)

Windows [Reading and writing shared data and log files authorized by an alternative global security group](#)

Windows [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

Important: By default, computers beginning with Windows 10 version 1607 and Windows Server 2016 are more restrictive than earlier versions of Windows.

This change restricts clients allowed to make remote calls to the Security Accounts Manager (SAM) and could impact IBM MQ with queue managers failing to start. Access to SAM is critical for the functioning of IBM MQ when IBM MQ is configured as a domain account.

1. Do the steps in [“Creating an Active Directory and DNS domain on Windows”](#) on page 548 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 544.

For the purposes of the task the domain controller host name 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 IBM WebSphere MQ 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 on Windows" on page 548, 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, see [Configuring IBM MQ with the Prepare IBM MQ Wizard](#).

- a) The IBM MQ Installer runs the Prepare IBM MQ Wizard automatically.
To start the wizard manually, find the shortcut to the Prepare IBM MQ Wizard 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 574, 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 Windows”](#) on page 554, to configure a file share to store the data and log files of a multi-instance queue manager.

Related tasks

 [Creating an Active Directory and DNS domain on Windows](#)

 [Creating a shared directory for queue manager data and log files on Windows](#)

 [Reading and writing shared data and log files authorized by an alternative global security group](#)

Related reference

[User rights required for an IBM MQ Windows Service](#)

 [Creating a shared directory for queue manager data and log files on Windows](#)

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 on Windows”](#) on page 548, 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 544.

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 556](#).

Related tasks

-  [Creating an Active Directory and DNS domain on Windows](#)
-  [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 on Windows” on page 548.](#)
2. [“Installing IBM MQ on a server or workstation in a Windows domain” on page 551.](#)
3. [“Creating a shared directory for queue manager data and log files on Windows” on page 554.](#)

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 544.](#)

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 *wmqa*.

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, 2026.  ALL RIGHTS RESERVED.  
The MQ service for installation 'Installation1' ended successfully.
```

And:

```
5724-H72 (C) Copyright IBM Corp. 1994, 2026.  ALL RIGHTS RESERVED.  
The MQ service for installation 'Installation1' started successfully.
```

4. Create the queue manager.

```
crtmqm -a wmqa\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, 2026.  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 AMQSGETO start
message A test message
Wait 15 seconds ...
no more messages
Sample AMQSGETO 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 on Windows](#)

 [Installing IBM MQ on a server or workstation in a Windows domain](#)

 [Creating a shared directory for queue manager data and log files on Windows](#)

 [Creating a multi-instance queue manager on Windows 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 [“Creating a multi-instance queue manager on domain workstations or servers on Windows”](#) on page 545.

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 AIX and Linux. On Windows, the checks are always successful.
2. Do the steps in “[Creating an Active Directory and DNS domain on Windows](#)” on page 548 to create the first domain controller.
3. Do the steps in “[Adding a second Windows domain controller to an example domain](#)” on page 563 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 Windows domain controllers in an example domain](#)” on page 564 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:

```
dspm
```

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 [“Verifying the multi-instance queue manager on Windows” on page 566](#).

Related tasks

[“Adding a second Windows domain controller to an example domain” on page 563](#)

[“Installing IBM MQ on Windows domain controllers in an example domain” on page 564](#)

Related information

[Windows 2000, Windows Server 2003, and Windows Server 2008 cluster nodes as domain controllers](#)

Windows *Adding a second Windows domain controller to an example 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 on Windows” on page 548](#) 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 544](#).

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 IBM WebSphere MQ 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 Windows domain controllers in an example domain”](#) on page 564.

Related tasks

 **Windows** [Installing IBM MQ on Windows domain controllers in an example domain “Creating an Active Directory and DNS domain on Windows” on page 548](#)

 **Windows** [Installing IBM MQ on Windows domain controllers in an example domain](#)
Install and configure installations of IBM MQ on both domain controllers in the *wmq.example.com* domain.

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 on Windows”](#) on page 548 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 Windows domain controller to an example domain”](#) on page 563 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, 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, see [Configuring IBM MQ with the Prepare IBM MQ Wizard](#).

- a) The IBM MQ installer runs the Prepare IBM MQ Wizard automatically.

To start the wizard manually, find the shortcut to the Prepare IBM MQ Wizard 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 Wizard 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 **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 574, to verify that the installation and configuration are working correctly.
2. Go back to the task, "[Creating a multi-instance queue manager on Windows domain controllers](#)" on page 559, to complete the task of configuring a multi-instance queue manager on domain controllers.

Related tasks

 [Adding a second Windows domain controller to an example domain](#)

Related reference

[User rights required for an IBM MQ Windows Service](#)

 [Verifying 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 [“Creating a multi-instance queue manager on Windows domain controllers”](#) on page 559. 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.



Attention: 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) +
CONNNAME(' 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 amqsgshac 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:\>amqsgshac 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

```

Windows *Securing 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 70 on page 570](#). 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.

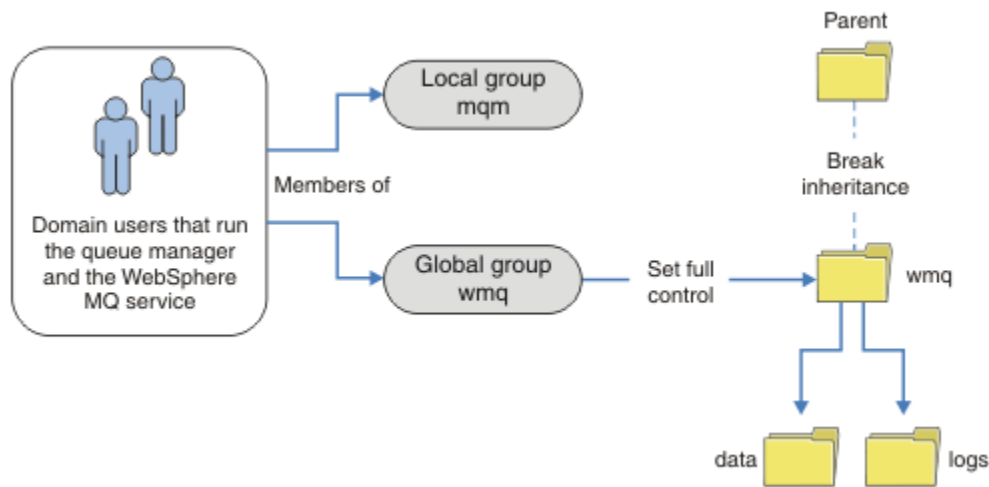


Figure 70. Securing queue manager data and logs using an alternative global security group (1)

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 70 on page 570](#), 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 70 on page 570](#) 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 71 on page 570](#). 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.

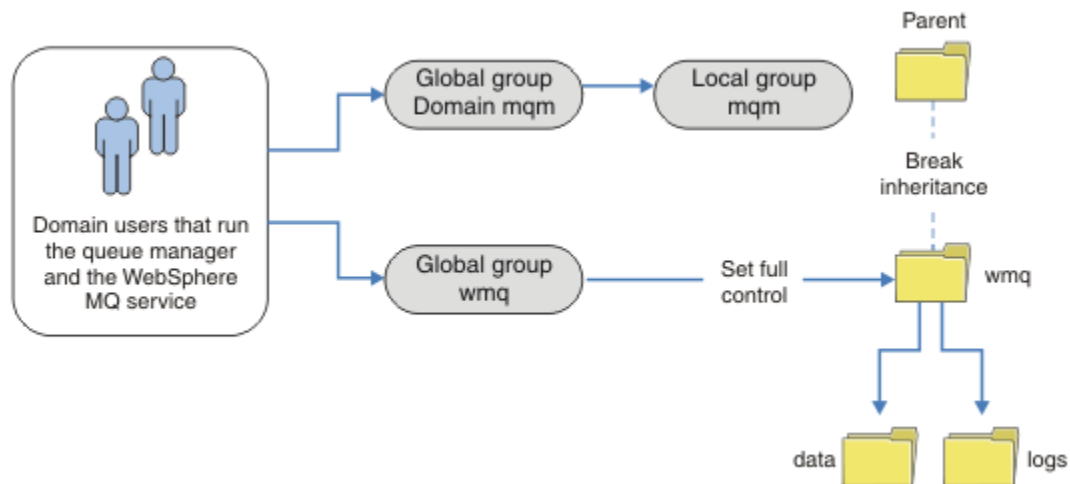


Figure 71. Securing queue manager data and logs using an alternative global security group (2)

The organization in Figure 71 on page 570 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 72 on page 571.

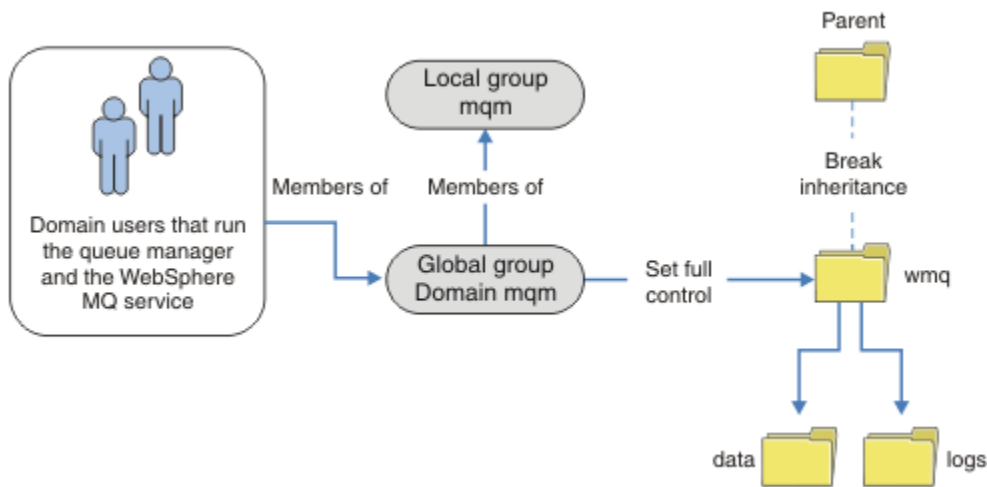


Figure 72. Securing queue manager data and logs using an alternative global security group (3)

Alternatively, you might need a finer degree of access control, with different queue managers restricted to being able to access different folders; see Figure 73 on page 572. In Figure 73 on page 572, 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 72 on page 571, because there are two groups of users.

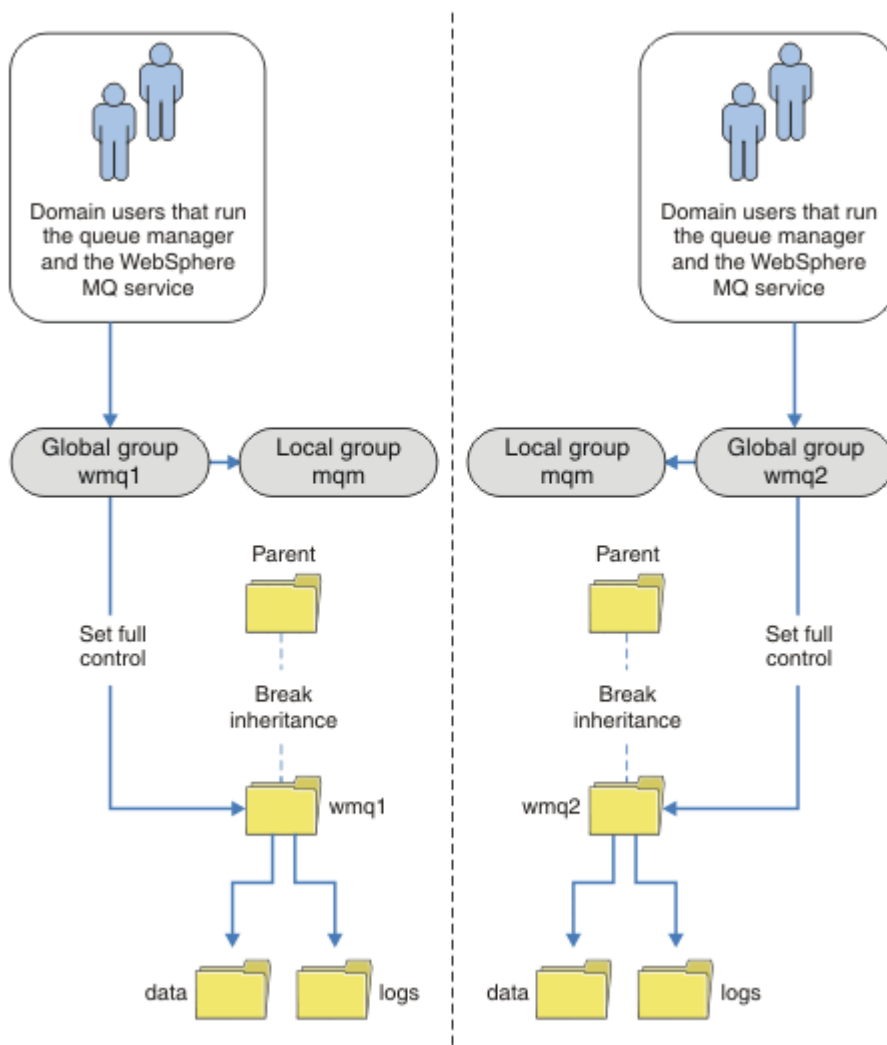


Figure 73. Securing queue manager data and logs using an alternative global security principal (4)

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 72 on page 571](#).

Windows 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 574](#), 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 577](#), 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 [“Securing shared queue manager data and log directories and files on Windows”](#) on page 569.

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 574, 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 74 on page 573](#). The local group, wmq, is an example of an alternative security group.

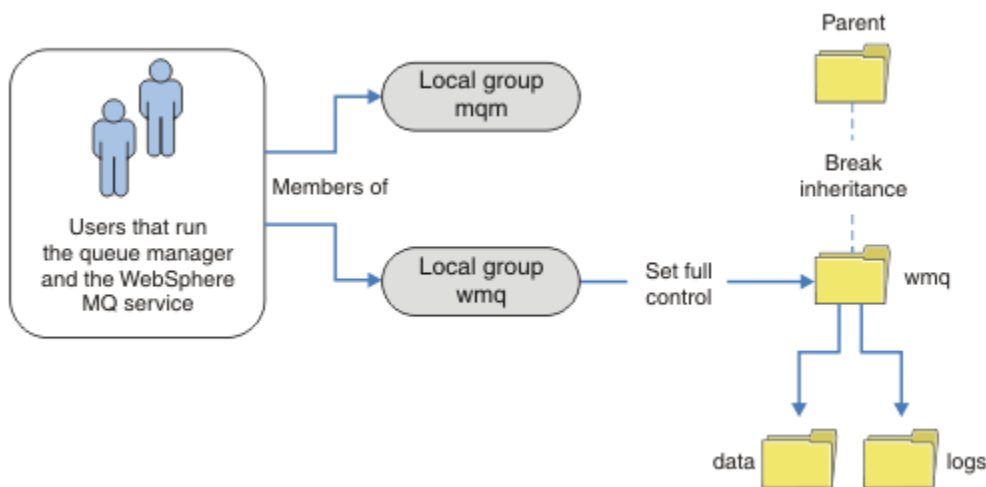


Figure 74. Securing queue manager data and logs using an alternative local security group, wmq

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 [“Securing shared queue manager data and log directories and files on Windows”](#) on page 569.
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 further information, see [Configuring IBM MQ with 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 544, 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 544.

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:

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 544](#)

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

Windows *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 556](#)

[“Creating a multi-instance queue manager on domain workstations or servers on Windows” on page 545](#)

Windows *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 further information, see [Configuring IBM MQ with 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 544](#), 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 544](#).

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 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 “Creating a multi-instance queue manager on domain workstations or servers on Windows” on page 545, the domain user IDs created in “Creating an Active Directory and DNS domain on Windows” on page 548 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, 2026. ALL RIGHTS RESERVED.  
The MQ service for installation 'Installation1' ended successfully.
```

And:

```
5724-H72 (C) Copyright IBM Corp. 1994, 2026. 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, 2026. 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

Windows

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.

Linux

Creating a multi-instance queue manager on Linux

An example showing how to set up a multi-instance queue manager on Linux. The setup is small to illustrate the concepts involved. The steps differ on other UNIX platforms.

Before you begin

This example is based on Red Hat Enterprise Linux 5 and Windows 7. Current versions of IBM MQ require later versions of both these products. However, the setup steps are still broadly the same as those described here.

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 Red Hat Enterprise Linux 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. An IBM 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. Read [IBM MQ installation overview](#) and follow the appropriate link 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 /MQHA *(rw, sync, no_wdelay, fsid=0) to /etc/exports

Note: The '*' allows all machines that can reach this one mount /MQHA for read/write. Restrict access on a production machine.
- 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](#).

Related reference

[IBM MQ file permissions in /opt/mqm with setuid for mqm](#)

Linux *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 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 “[Creating a multi-instance queue manager on Linux](#)” on page 581. 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 TCP/IP 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:\>amqsgshac 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

```

Converting a single instance to a multi-instance queue manager on AIX and Linux

To convert a single instance queue manager to a multi-instance queue manager on AIX and Linux, you must move the queue manager data to a shared directory, and reconfigure the queue manager on two other servers.

Before you begin

You must check the prerequisites for running a multi-instance queue manager as part of this task. For a list of tested environments, see [Testing statement for IBM MQ multi-instance queue manager file systems](#). Other environments might work; a test tool is provided with IBM MQ to assist you in qualifying other environments.

You must have three servers to run a multi-instance queue manager. One server has a shared file system to store the queue manager data and logs. The other servers run the active and standby instances of the queue manager.

About this task

You have a single-instance queue manager that you want to convert to a multi-instance queue manager. The queue manager conversion itself is straightforward, but you must do other tasks to create a fully automated production environment.

You must check the prerequisites for a multi-instance queue manager, set up the environment and check it. You must set up a monitoring and management system to detect if the multi-instance queue manager has failed and been automatically restarted. You can then find out what caused the restart, remedy it, and restart the standby. You must also modify applications, or the way applications are connected to the queue manager, so that they can resume processing after a queue manager restart.

Procedure

1. Check the operating system that you are going to run the queue manager on, and the file system on which the queue manager data and logs are stored on. Check that they can run a multi-instance queue manager.
 - a) Consult [Testing statement for IBM MQ multi-instance queue manager file systems](#). See whether the combination of operating system and file system is tested and capable of running a multi-instance queue manager.

A shared file system must provide lease-based locking to be adequate to run multi-instance queue managers. Lease-based locking is a recent feature of some shared file systems, and in some case fixes are required. The support statement provides you with the essential information.
 - b) Run **amqmfscck** to verify that the file system is configured correctly.

File systems are sometimes configured with performance at a premium over data integrity. It is important to check the file system configuration. A negative report from the **amqmfscck** tool

tells you the settings are not adequate. A positive result is an indication that the file system is adequate, but the result is not a definitive statement that the file system is adequate. It is a good indication.

- c) Run the integrity checking application provided in the technote, [Testing a shared file system for compatibility with IBM MQ Multi-instance Queue Managers](#).

The checking application tests that the queue manager is restarting correctly.

2. Configure a user and group to be able to access a share on the networked file system from each server that is running a queue manager instance.

On AIX and Linux, the `uid` and `gid` for `mqm` in `/etc/passwd` must be the same on each system; see [Creating a multi-instance queue manager on Linux](#).

3. Set up a directory for the share on the networked file system with the correct access permissions.

A typical configuration is to set up a single shared directory that contains all data and log directories for all queue managers that use the shared disk; see [Share named qmgrs and log directories in Example directory configurations on AIX and Linux systems](#).

For example, create a root directory on the share called `MQHA` that has subdirectories `data` and `logs`. Each queue manager creates its own data and log directories under `data` and `logs`. Create `/MQHA` on the shared drive. `/MQHA` is owned by the user and group `mqm` and has the access permissions `rxw`.

4. Copy the queue manager data and the logs to the share.

Follow the procedure to back up the queue manager described in [Backing up queue manager data](#).

Note: Unlike on Windows, the `hamvmqm` utility cannot be used on AIX and Linux.

5. Update the queue manager configuration information stored on the current queue manager server by completing the following steps:

- a) Modify the `Log:` stanza in the queue manager `qm.ini` file, which is on the *share*:

```
LogPath= share/logs/QMgrName
```

- b) Modify the `QueueManager:` stanza in the IBM MQ `mqs.ini` file, which is typically in the `/var/mqm` directory on AIX and Linux:

```
DataPath= share/data/QMgrName
```

where *QMGrName* is the Directory name in the `QueueManager:` stanza in the `mqs.ini` file and *share* is share where the data and logs are moved to.

6. Add the queue manager configuration information to the new queue manager server.

- a) Run the **dspmqinf** command to display the queue manager information.

Run the command on the server that ran the queue manager.

```
dspmqinf -o command QMgrName
```

The command output is formatted ready to create a queue manager configuration.

```
addmqinf -s QueueManager -v Name= QMgrName -v Directory= QMgrName -v  
Prefix=d:\var\mqm Datapath= \share\data\QMGrName
```

- b) Create a queue manager configuration on the other server.

Run the **addmqinf** command copied from the previous output.

7. Add the network address of the new server to the connection name in client and channel definitions.

- a) Find all the client, sender, and requester TCP/IP settings that refer to the server.

Client settings might be in Client Definition Tables (CCDT), in environment variables, in Java properties files, or in client code. Cluster channels automatically discover the connection name of a queue manager from its cluster receiver channel. As long as the cluster receiver channel name is blank or omitted, TCP/IP discovers the IP address of the server hosting the queue manager.

- b) Modify the connection name for each of these connections to include the TCP/IP addresses of both servers that are hosting the multi-instance queue manager.

For example, change the following connection name:

```
echo DISPLAY CHANNEL(ENGLAND) CONNAME | runmqsc QM1
```

```
5724-H72 (C) Copyright IBM Corp. 1994, 2026. ALL RIGHTS RESERVED.  
Starting MQSC for queue manager QM1.  
1: DISPLAY CHANNEL(ENGLAND) CONNAME  
AMQ8414: Display Channel details.  
CHANNEL(ENGLAND) CHLTYPE(SDR)  
CONNAME(LONDON)
```

into:

```
echo ALTER CHANNEL(ENGLAND) CHLTYPE(SDR) CONNAME('LONDON, BRISTOL') | runmqsc QM1
```

8. Update your monitoring and management procedures to detect the queue manager restarting.
9. Update client applications to be automatically reconnectable, if appropriate.
10. Update the start procedure for your IBM MQ applications to be started as queue manager services.
11. Start each instance of the queue manager, permitting them to be highly available.

The first instance of the queue manager that is started becomes the active instance. Issue the command twice, once on each server.

```
strmqm -x QMgrName
```

What to do next

To get the highest availability out of multi-instance queue managers, you must design client applications to be reconnectable and server applications to be restartable; see [Application recovery](#).

Related concepts

[Application recovery](#)

[Automatic client reconnection](#)

[Channel and client reconnection](#)

[Multi-instance queue managers](#)

[Queue manager configuration files, qm.ini](#)

[Shared file system](#)

Related tasks

[Backing up queue manager data](#)

[Changing IBM MQ configuration information on Multiplatforms](#)

[Creating a multi-instance queue manager on Linux](#)

[Moving a queue manager to MSCS storage](#)

[Verifying shared file system locking](#)

Related reference

[amqmfsc \(file system check\)](#)

[The IBM MQ configuration file, mqs.ini](#)

Related information

[Testing a shared file system for compatibility with IBM MQ Multi-instance Queue Managers](#)

[Testing statement for IBM MQ multi-instance queue manager file systems](#)

Converting a single instance to a multi-instance queue manager on Windows

To convert a single instance queue manager, to a multi-instance queue manager, on Windows platforms, you must move the queue manager data to a shared directory, and reconfigure the queue manager on two other servers.

Before you begin

You must check the prerequisites for running a multi-instance queue manager as part of this task. For a list of tested environments, see [Testing statement for IBM MQ multi-instance queue manager file systems](#). Other environments might work; a test tool is provided with IBM MQ to assist you in qualifying other environments.

You must have three servers to run a multi-instance queue manager. One server has a shared file system to store the queue manager data and logs. The other servers run the active and standby instances of the queue manager.

About this task

You have a single-instance queue manager that you want to convert to a multi-instance queue manager. The queue manager conversion itself is straightforward, but you must do other tasks to create a fully automated production environment.

You must check the prerequisites for a multi-instance queue manager, set up the environment and check it. You must set up a monitoring and management system to detect if the multi-instance queue manager has failed and been automatically restarted. You can then find out what caused the restart, remedy it, and restart the standby. You must also modify applications, or the way applications are connected to the queue manager, so that they can resume processing after a queue manager restart.

Procedure

1. Check the operating system that you are going to run the queue manager on, and the file system on which the queue manager data and logs are stored on. Check that they can run a multi-instance queue manager.
 - a) Consult [Testing statement for IBM MQ multi-instance queue manager file systems](#). See whether the combination of operating system and file system is tested and capable of running a multi-instance queue manager.

A shared file system must provide lease-based locking to be adequate to run multi-instance queue managers. Lease-based locking is a recent feature of some shared file systems, and in some case fixes are required. The support statement provides you with the essential information.

- b) Run the integrity checking application provided in the technote, [Testing a shared file system for compatibility with IBM MQ Multi-instance Queue Managers](#).
The checking application tests that the queue manager is restarting correctly.
2. Configure a user and group to be able to access a share on the networked file system from each server that is running a queue manager instance.
On Windows, the security IDs (SIDs) of the mqm group can be different; see [Windows domains and multi-instance queue managers](#).
3. Set up a directory for the share on the networked file system with the correct access permissions.
A typical configuration is to set up a single shared directory that contains all data and log directories for all queue managers that use the shared disk; see [Share named qmgrs and log directories](#).
For example, create a root directory on the share called MQHA that has subdirectories data and logs. Each queue manager creates its own data and log directories under data and logs. Create *drive* \MQHA on the shared drive. The owner is a member of mqm. mqm must have full-control authority. Create a share for *drive* \MQHA.
4. Copy the queue manager data and the logs to the share.
On Windows, you can run the **hamvmqm** command to move the queue manager data to the share.
5. Add the queue manager configuration information to the new queue manager server.
 - a) Run the **dspmqinf** command to display the queue manager information
Run the command on the server that ran the queue manager.

```
dspmqinf -o command QMgrName
```


The command output is formatted ready to create a queue manager configuration.

```
addmqinf -s QueueManager -v Name= QMgrName -v Directory= QMgrName -v  
Prefix=d:\var\mqm Datapath= \share\data\QMgrName
```

- b) Create a queue manager configuration on the other server.
Run the **addmqinf** command copied from the previous output.
6. Add the network address of the new server to the connection name in client and channel definitions.
 - a) Find all the client, sender, and requester TCPIP settings that refer to the server.
 - Client settings might be in Client Definition Tables (CCDT), in environment variables, in Java properties files, or in client code.
 - Cluster channels automatically discover the connection name of a queue manager from its cluster receiver channel. As long as the cluster receiver channel name is blank or omitted, TCPIP discovers the IP address of the server hosting the queue manager.
 - b) Modify the connection name for each of these connections to include the TCPIP addresses of both servers that are hosting the multi-instance queue manager.
For example, change the following connection name:

```
echo DISPLAY CHANNEL(ENGLAND) CONNAME | runmqsc QM1
```

```
5724-H72 (C) Copyright IBM Corp. 1994, 2026. ALL RIGHTS RESERVED.  
Starting MQSC for queue manager QM1.  
1: DISPLAY CHANNEL(ENGLAND) CONNAME  
AMQ8414: Display Channel details.  
CHANNEL(ENGLAND) CHLTYPE(SDR)  
CONNAME(LONDON)
```

into:

```
echo ALTER CHANNEL(ENGLAND) CHLTYPE(SDR) CONNAME('LONDON, BRISTOL') | runmqsc QM1
```

7. Update your monitoring and management procedures to detect the queue manager restarting.
8. Update client applications to be automatically reconnectable, if appropriate.
9. Update the start procedure for your IBM MQ applications to be started as queue manager services.
10. Start each instance of the queue manager, permitting them to be highly available.

The first instance of the queue manager that is started becomes the active instance. Issue the command twice, once on each server.

```
strmqm -x QMgrName
```

What to do next

To get the highest availability out of multi-instance queue managers, you must design client applications to be reconnectable and server applications to be restartable; see [Application recovery](#).

Related concepts

[Application recovery](#)

[Automatic client reconnection](#)

[Channel and client reconnection](#)

[Multi-instance queue managers](#)

[Queue manager configuration files, qm.ini](#)

[Shared file system](#)

[Windows domains and multi-instance queue managers](#)

Related tasks

[Backing up queue manager data](#)

[Changing IBM MQ configuration information on Multiplatforms](#)

[Moving a queue manager to MSCS storage](#)

[Verifying shared file system locking](#)

[Working with services](#)

Related reference

amqmfsc (file system check)

Related information

[Testing a shared file system for compatibility with IBM MQ Multi-instance Queue Managers](#)

[Testing statement for IBM MQ multi-instance queue manager file systems](#)

Deleting a multi-instance queue manager

On Multiplatforms, to delete a multi-instance queue manager completely, you 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 AIX 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
```

Reverting to a single-instance queue manager on AIX and Linux

Revert a multi-instance queue manager to a single instance queue manager, on AIX and Linux, by stopping the standby instance. Then restart the active instance and do not set the flag that permits standby instances.

Before you begin

You have at least three servers configured to run a queue manager as a multi-instance queue manager. The queue manager is currently running as a multi-instance queue manager, with one standby instance active.

About this task

The task involves deactivating the active standby so that only the running multi-instance queue manager remains active. To prevent a standby instance being started in the future, you must stop the active instance and restart it. When you restart it, you start it as a single instance queue manager that prevents standby instances being started. The standby instance is stopped as a separate step, to give you the option of restarting the active instance at a later date. You can stop both instances by running the standard `endmqm QMgrName` command on the server running the active queue manager.

Procedure

1. Stop the standby queue manager instance.

On the server running the standby instance:

```
endmqm -w QMgrName
```

2. Stop the active queue manager instance.

On the server running the active instance:

```
endmqm -w (QMgrName)
```

3. Restart the queue manager, preventing standbys.

On the server that is going to run the queue manager:

```
strmqm QMgrName
```

What to do next

You might want to run the queue manager as a single instance on the same server as the queue manager data.

When the queue manager is stopped move the queue manager data back to the server that is running the queue manager. Alternatively install IBM MQ, and then move the queue manager configuration definition onto the server with the queue manager data. Both tasks are variations of steps in [“Converting a single instance to a multi-instance queue manager on AIX and Linux” on page 585](#) to create a multi-instance queue manager.

Reverting to a single-instance queue manager on Windows

Revert a multi-instance queue manager to a single instance queue manager, on Windows platforms, by stopping the standby instance. Then restart the active instance and do not set the flag that permits standby instances.

Before you begin

You have at least three servers configured to run a queue manager as a multi-instance queue manager. The queue manager is currently running as a multi-instance queue manager, with one standby instance active.

About this task

The task involves deactivating the active standby so that only the running multi-instance queue manager remains active. To prevent a standby instance being started in the future, you must stop the active instance and restart it. When you restart it, you start it as a single instance queue manager that prevents standby instances being started. The standby instance is stopped as a separate step, to give you the option of restarting the active instance at a later date. You can stop both instances by running the standard `endmqm QMgrName` command on the server running the active queue manager.

Procedure

1. Stop the standby queue manager instance.

On the server running the standby instance:

```
endmqm -w QMgrName
```

2. Stop the active queue manager instance.

On the server running the active instance:

```
endmqm -w (QMgrName)
```

3. Restart the queue manager, preventing standbys.

On the server that is going to run the queue manager:

```
strmqm QMgrName
```

What to do next

You might want to run the queue manager as a single instance on the same server as the queue manager data.

When the queue manager is stopped move the queue manager data back to the server that is running the queue manager. Alternatively install IBM MQ, and then move the queue manager configuration definition onto the server with the queue manager data. Both tasks are variations of steps in [Converting a single instance to a multi-instance queue manager on Windows](#).

Starting and stopping a multi-instance queue manager

Starting and stopping a queue manager configured on Multiplatforms 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

On Multplatforms, 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.

Multi 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.

Multi 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 TCP/IP, 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..

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 597.

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 `MQCONNEX` 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 `MQCONNEX` 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 596, 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.

Note that for a call back, it might be necessary to resume the consuming application if the call back consumer parameter state is: `MQCS_SUSPENDED_USER_ACTION`.

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 (which may also suspend asynchronous consumers, see the preceding text).
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.

Note: In a Native HA configuration, any change of leader of a queue manager instance always results in the discarding of non-persistent messages regardless of the setting of **NPMCLASS**.

- 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 MQCOMMIT 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 re-create 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 AIX, Linux, and Windows 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 AIX 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.

External High Availability solutions

In external HA solutions, availability is provided by an HA infrastructure external to IBM MQ that reinstates the queue manager following a failure. It is dependent on replicated storage that can be re-mounted in the new location.

HA clusters on AIX and Linux

You can use IBM MQ with a high availability (HA) cluster on AIX 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.

This section introduces [“HA cluster configurations”](#) on page 600, the relationship of HA clusters to queue manager clusters, [“IBM MQ clients”](#) on page 601, and [“IBM MQ operating in an HA cluster”](#) on page 601, 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 MQ includes 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.

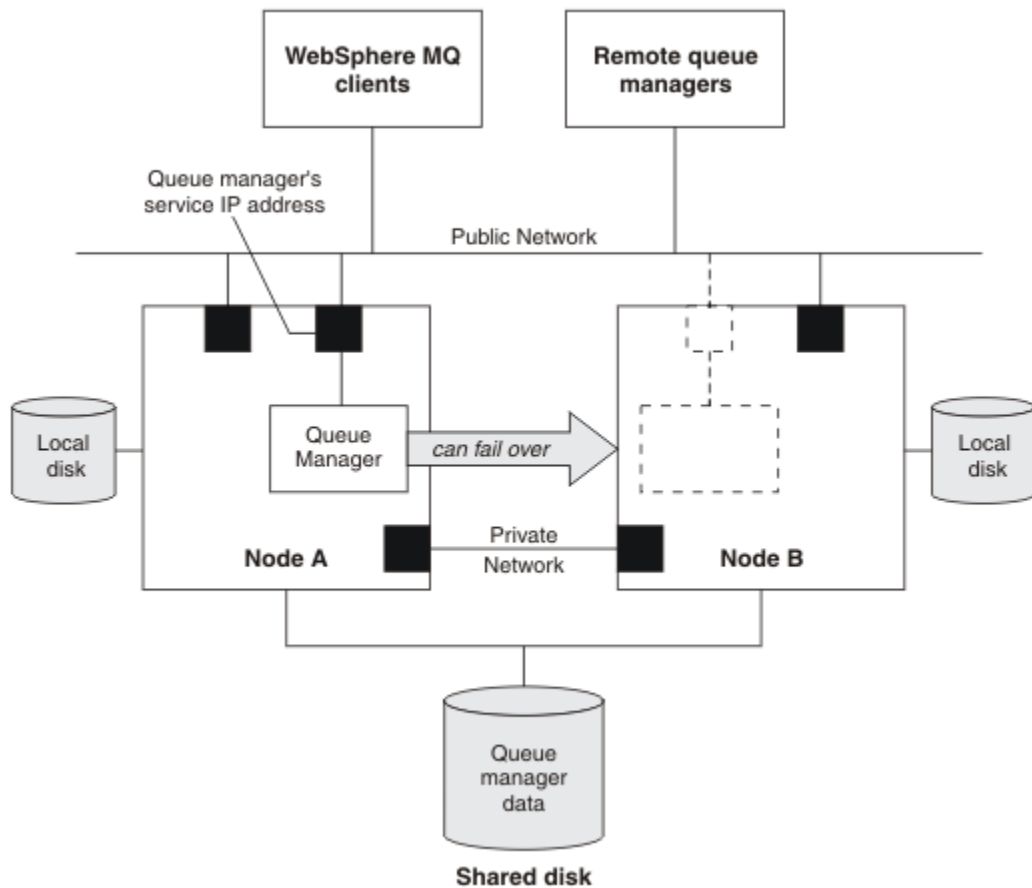


Figure 75. HA cluster

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.

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.

About this task

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.

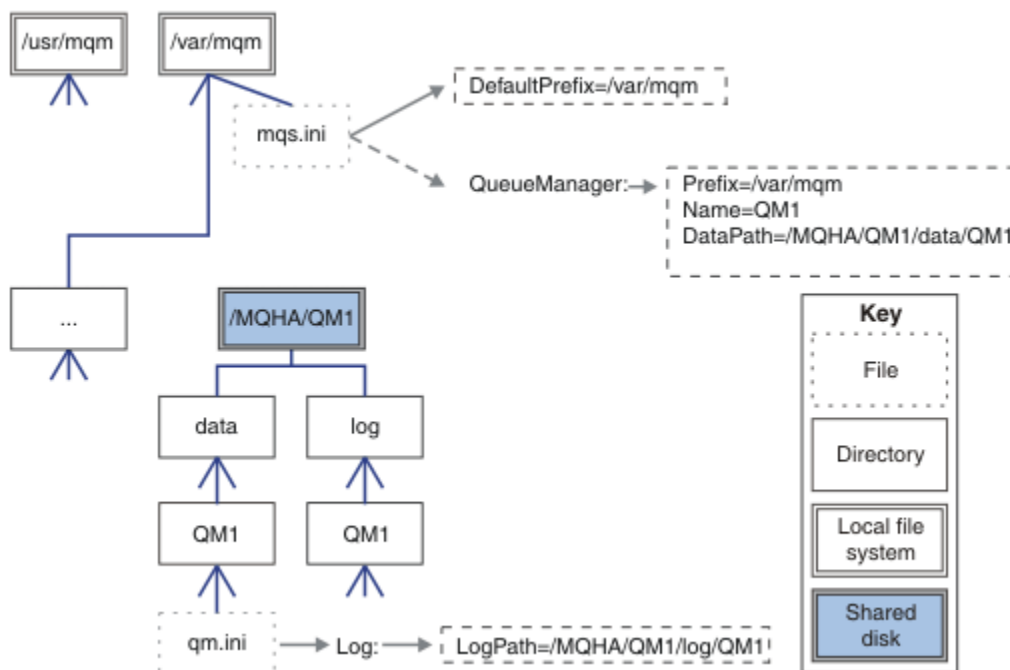


Figure 76. Shared named data and log directories

Procedure

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.

Creating an HA cluster queue manager on AIX and Linux

The first step towards using a queue manager in a high availability cluster is to create the queue manager on one of the nodes.

About this task

To create a queue manager for use in an HA cluster, you must first select one of the nodes in the cluster on which to create the queue manager, and then complete the following steps on this node.

Procedure

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:

```
dspmqinf -o command qmgrname
```

where *qmgrname* is the name of the queue manager.

For more information about using the **addmqinf** command, see [“Adding queue manager configuration to other HA cluster nodes on AIX and Linux” on page 604](#).

The **addmqinf** command is displayed in a similar way to the following example:

```
addmqinf -sQueueManager -vName=qmgrname -vDirectory=qmgrname \  
-vPrefix=/var/mqm -vDataPath=/MQHA/qmgrname/data/qmgrname
```

7. Make a careful note of the displayed command.
8. Unmount the queue manager's file systems.

What to do next

You are now ready to complete the steps described in [“Adding queue manager configuration to other HA cluster nodes on AIX and Linux” on page 604](#).

Adding queue manager configuration to other HA cluster nodes on AIX and Linux

You must add the queue manager configuration information to the other nodes in the HA cluster.

Before you begin

Before you complete this task, you must have completed the steps in [“Creating an HA cluster queue manager on AIX and Linux” on page 604](#). After you have created the queue manager, you must then add the configuration information for the queue manager to each of other nodes in the HA cluster by completing the following steps on each of the other nodes.

About this task

When you create a queue manager for use in an HA cluster, you must first select one of the nodes in the cluster on which to create the queue manager, as described in [“Creating an HA cluster queue manager on AIX and Linux” on page 604](#).

Procedure

1. Mount the queue manager file systems.
2. Add the queue manager configuration information to the node.

There are two ways of adding the configuration information:

- By editing `/var/mqm/mqs.ini` directly.
- By issuing the **addmqinf** command that was displayed by the **dspmqinf** command in step 6 in [“Creating an HA cluster queue manager on AIX and Linux” on page 604](#).

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 from the one that is currently associated with the queue manager, 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.

Example shell scripts for starting an HA cluster queue manager on AIX and Linux

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 amqirmfa amqimppa \
               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.

Example shell script for stopping an HA cluster queue manager on AIX and Linux

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 a queue manager 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 [“Example shell scripts for starting an HA cluster queue manager on AIX and Linux”](#) on page 605.)

```

#!/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 amqfcpub 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 *.*$"
        cnt=`ps -ef | tr "\t" " " | grep amqzma0 | grep -v grep | \
            egrep "$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 amqzma0 amqfcxba amqfpub amqpcsea amqzlaa0 \
    amqzlsa0 runmqchi runmqslr amqcrsta amqirmfa amqimppa \
    amqzfuma amqmuf0 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 an HA cluster queue manager on AIX and Linux

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.

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 you might need to perform some additional steps. If you have a primary installation you do not need to specify the `MQ_INSTALLATION_PATH` to use the script. Otherwise, use the following steps ensure that the `MQ_INSTALLATION_PATH` is identified correctly:

1. Use the **crtmqenv** command from an IBM MQ 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 [“Example shell scripts for starting an HA cluster queue manager on AIX and Linux” on page 605](#)):

```
#!/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 HA cluster control on AIX and Linux

You must configure the queue manager, under control of the HA cluster, with the queue manager's IP address and shared disks.

About this task

To put the queue manager under control of the HA cluster, you must define a resource group to contain the queue manager and all of its associated resources.

Procedure

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 an HA cluster queue manager on AIX and Linux

You might want to remove a queue manager from a node that is no longer required to run the queue manager.

About this task

To remove the queue manager from a node in an HA cluster, you must remove its configuration information.

Procedure

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`
3. Optional: To completely delete the queue manager, use the **dlmqm** command.

Important: Be aware that deleting the queue manager by using the **dlmqm** command 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 Microsoft Cluster Service (MSCS)

Introducing and setting up MSCS to support failover of virtual servers. MSCS is also known as Windows Server Failover Clustering (WSFC).

This information applies to IBM MQ for Windows only.

Note: From Windows Server 2016, the new name for Microsoft Cluster Service (MSCS) is Windows Server Failover Clustering (WSFC).

MSCS/WSFC enables you to connect servers into a cluster, giving higher availability of data and applications, and making it easier to manage the system. MSCS/WSFC can automatically detect and recover from server or application failures.

MSCS/WSFC supports failover of virtual servers, which correspond to applications, websites, print queues, or file shares (including, for example, their disk spindles, files, and IP addresses).

Failover is the process by which MSCS/WSFC 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 re-initiates the application there.

For information about how to configure and use failover clusters, see the subtopics.

Introducing MSCS clusters

Microsoft Cluster Service (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.

Note: From Windows Server 2016, the new name for Microsoft Cluster Service (MSCS) is Windows Server Failover Clustering (WSFC).

For information about the various high availability options, such as MSCS clusters, multi-instance queue managers, and IBM MQ clusters, see [“Configuring high availability and disaster recovery”](#) on page 477.

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 77](#) on page 611.

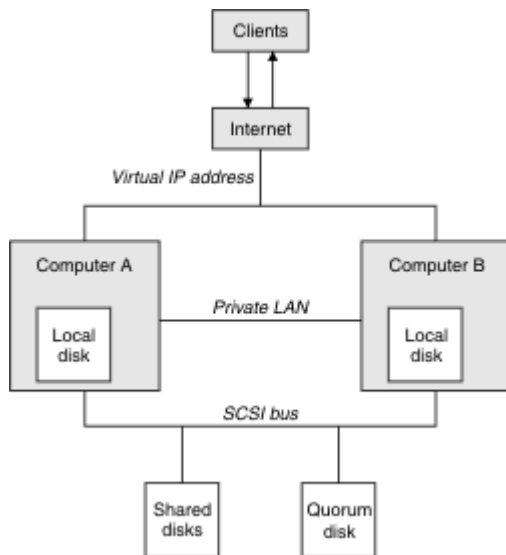


Figure 77. Two-computer MSCS cluster

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.

Windows 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.

Note: From Windows Server 2016, the new name for Microsoft Cluster Service (MSCS) is Windows Server Failover Clustering (WSFC).

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 614. If you want to create new queue managers for use with MSCS, see [“Creating a queue manager for use with MSCS”](#) on page 613.

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 615.

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 618.

Windows Setup symmetry and MSCS

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 612.

Windows 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.

Windows 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.

Windows Cluster modes and MSCS

There are two modes in which you might run a cluster system with IBM MQ on Windows: 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 Microsoft Cluster Service (MSCS) control.

Note: From Windows Server 2016, the new name for Microsoft Cluster Service (MSCS) is Windows Server Failover Clustering (WSFC).

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:\IBM\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 614.

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 614](#).

Windows

Moving a queue manager to MSCS storage

This procedure configures an existing queue manager to make it suitable for putting under Microsoft Cluster Service (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.



Attention: 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 TLS connections and the TLS 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 TLS 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 TLS key repository file.

Note: From Windows Server 2016, the new name for Microsoft Cluster Service (MSCS) is Windows Server Failover Clustering (WSFC).

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 615](#).
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 620](#) 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 615](#).

Putting a queue manager under MSCS control

How to place a queue manager under Microsoft Cluster Service (MSCS) control, including prerequisite tasks.

Note: From Windows Server 2016, the new name for Microsoft Cluster Service (MSCS) is Windows Server Failover Clustering (WSFC).

Before you put a queue manager under MSCS/WSFC control

Before you put a queue manager under MSCS/WSFC control, complete the following steps:

1. Ensure that IBM MQ and its MSCS/WSFC 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 611](#).
2. Use the **haregtyp** utility program to register IBM MQ as an MSCS resource type on all the cluster nodes. See [“Support for MSCS utility programs” on page 622](#).
3. If you have not yet done so, [create a queue manager for use with MSCS/WSFC](#).
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 614](#).
5. If the queue manager is running, stop it by using either a command prompt or the IBM MQ Explorer.
6. Test MSCS/WSFC operation of the shared drives before going on to any of the following Windows procedures in this topic.

Windows Server 2019 or 2022

To place a queue manager under MSCS/WSFC control on Windows Server 2019 or 2022, 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 WSFC 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 WSFC 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 IBM 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 in MSCS” on page 621.](#)
- 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 in MSCS” on page 621.](#)

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 [“looksAlive and isAlive polling on MSCS”](#) on page 617.

- d. Optionally, set a preferred node (but note the comments in [“Using preferred nodes in MSCS”](#) on page 621)
 - 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 614.) You can also right-click and select **Initiate Failure**; the action (local restart or failover) depends on the current state and the configuration settings.

looksAlive and isAlive polling on MSCS

looksAlive and *isAlive* are intervals at which Microsoft Cluster Service (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.

Note: From Windows Server 2016, the new name for Microsoft Cluster Service (MSCS) is Windows Server Failover Clustering (WSFC).

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 dependent resources to another node in the cluster. For further information, see [MSCS documentation](#).

Windows **Removing a queue manager from MSCS control**

You can remove queue managers from Microsoft Cluster Service (MSCS) control, and return them to manual administration.

Note: From Windows Server 2016, the new name for Microsoft Cluster Service (MSCS) is Windows Server Failover Clustering (WSFC).

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 TLS connections you must modify the queue manager attribute, SSLKEYR, using the IBM MQ Explorer or the MQSC command ALTER QMGR, to point to the TLS 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 618](#)
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 618](#).
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 620](#) 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 614).

Hints and tips on using MSCS

This section contains some general information to help you use IBM MQ support for Microsoft Cluster Service (MSCS) effectively.

Note: From Windows Server 2016, the new name for Microsoft Cluster Service (MSCS) is Windows Server Failover Clustering (WSFC).

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 613 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 and MSCS

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 614 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 614, 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 [“Support for MSCS utility programs” on page 622](#) 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 77 on page 611](#)). 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 613](#)
- [“Moving a queue manager to MSCS storage” on page 614](#)
- [“Putting a queue manager under MSCS control” on page 615](#)

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 614](#)).

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.

Windows *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.

Windows *Working in Active/Active mode in MSCS*

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 in MSCS

Use these commands to integrate IBM MQ MSCS support with other systems. You can use them to issue IBM MQ commands, with 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 in MSCS

It can be useful when using Active/Active mode in MSCS 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.

COM+ errors when you install on MSCS

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.

Windows **Support for MSCS utility programs**

A list of the IBM MQ support for Microsoft Cluster Service (MSCS) utility programs that you can run at a command prompt.

Note: From Windows Server 2016, the new name for Microsoft Cluster Service (MSCS) is Windows Server Failover Clustering (WSFC).

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 618](#).
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 614](#).

Delete a queue manager from a node

hadlrmqm.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:

```
hadlrmqm /m qmname
```

where qmname is the name of the queue manager to remove.

Check and save setup details

amqmsyn.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.

Linux **MQ Adv. RDQM high availability**

RDQM (replicated data queue manager) is a high availability solution that is available on Red Hat Enterprise Linux for x86-64 platforms.

An RDQM configuration consists of three servers configured in a high availability (HA) group, each with an instance of the queue manager. One instance is the running queue manager, which synchronously replicates its data to the other two instances. If the server running this queue manager fails, another instance of the queue manager starts and has current data to operate with. The three instances of the

queue manager can optionally share a floating IP address, so clients only need to be configured with a single IP address. Only one instance of the queue manager can run at any one time, even if the HA group becomes partitioned due to network problems. The server running the queue manager is known as the 'primary', each of the other two servers is known as a 'secondary'.

Three nodes are used to greatly reduce the possibility of a split-brain situation arising. In a two-node High Availability system split-brain can occur when the connectivity between the two nodes is broken. With no connectivity, both nodes could run the queue manager at the same time, accumulating different data. When connection is restored, there are two different versions of the data (a 'split-brain'), and manual intervention is required to decide which data set to keep, and which to discard.

RDQM uses a three node system with quorum to avoid the split-brain situation. Nodes that can communicate with at least one of the other nodes form a quorum. Queue managers can only run on a node that has quorum. The queue manager cannot run on a node which is not connected to at least one other node, so can never run on two nodes at the same time:

- If a single node fails, the queue manager can run on one of the other two nodes. If two nodes fail, the queue manager cannot run on the remaining node because the node does not have quorum (the remaining node cannot tell whether the other two nodes have failed, or they are still running and it has lost connectivity).
- If a single node loses connectivity, the queue manager cannot run on this node because the node does not have quorum. The queue manager can run on one of the remaining two nodes, which do have quorum. If all nodes lose connectivity, the queue manager is unable to run on any of the nodes, because none of the nodes have quorum.

Note: You can use both the IBM MQ Console and IBM MQ Explorer with replicated data queue managers, but they do not display information that is specific to the RDQM features.

The group configuration of the three nodes is handled by Pacemaker. The replication between the three nodes is handled by DRBD. (See <https://clusterlabs.org/pacemaker/> for information about Pacemaker and <https://docs.linbit.com/docs/users-guide-9.0/> for information about DRBD.)

You can back up your replicated data queue managers by using the process described in [“Backing up queue manager data”](#) on page 760. Stopping the queue manager and backing it up has no effect on the node monitoring done by the RDQM configuration.

The following figure shows a typical deployment with an RDQM running on each of the three nodes in the HA group.

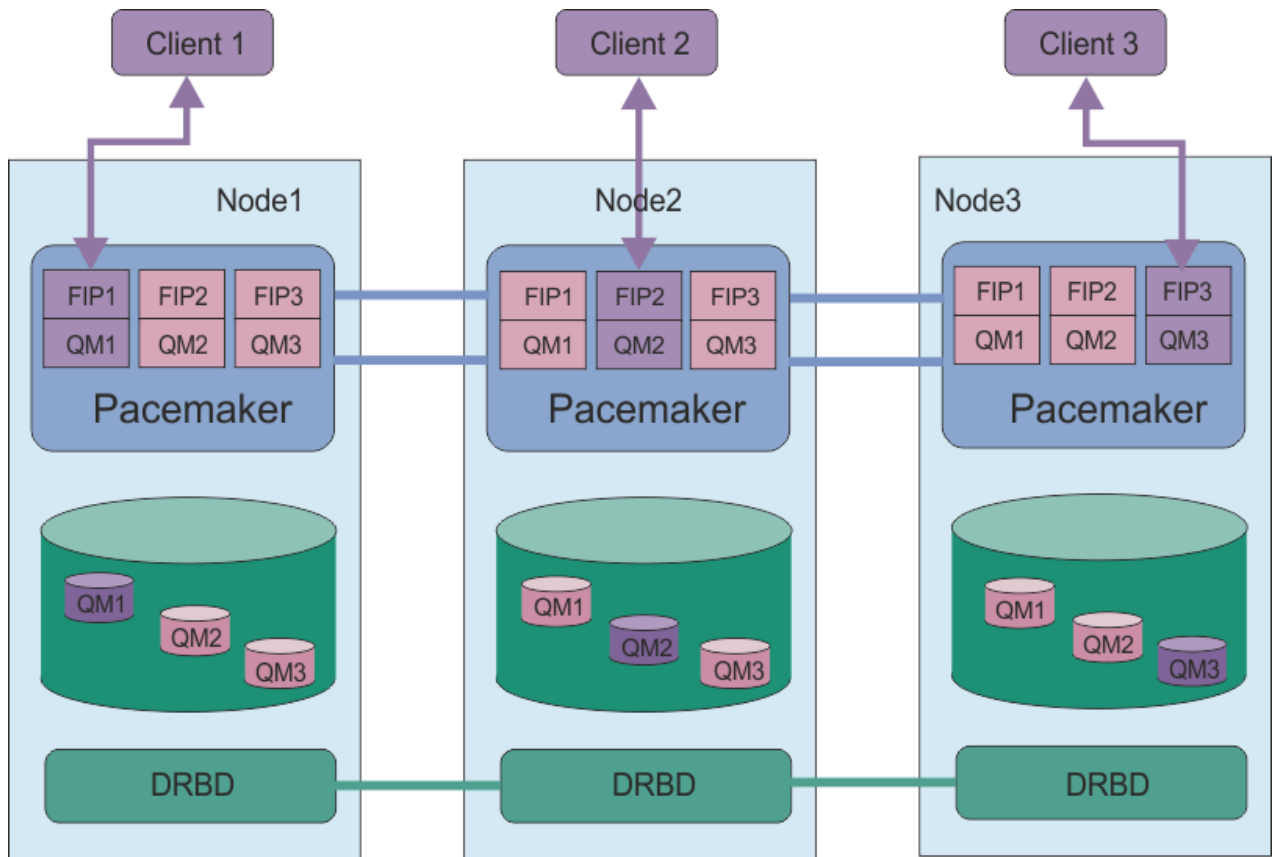


Figure 78. Example of HA group with three RDQMs

In the next figure, Node3 has failed, the Pacemaker links have been lost, and queue manager QM3 runs on Node2 instead.

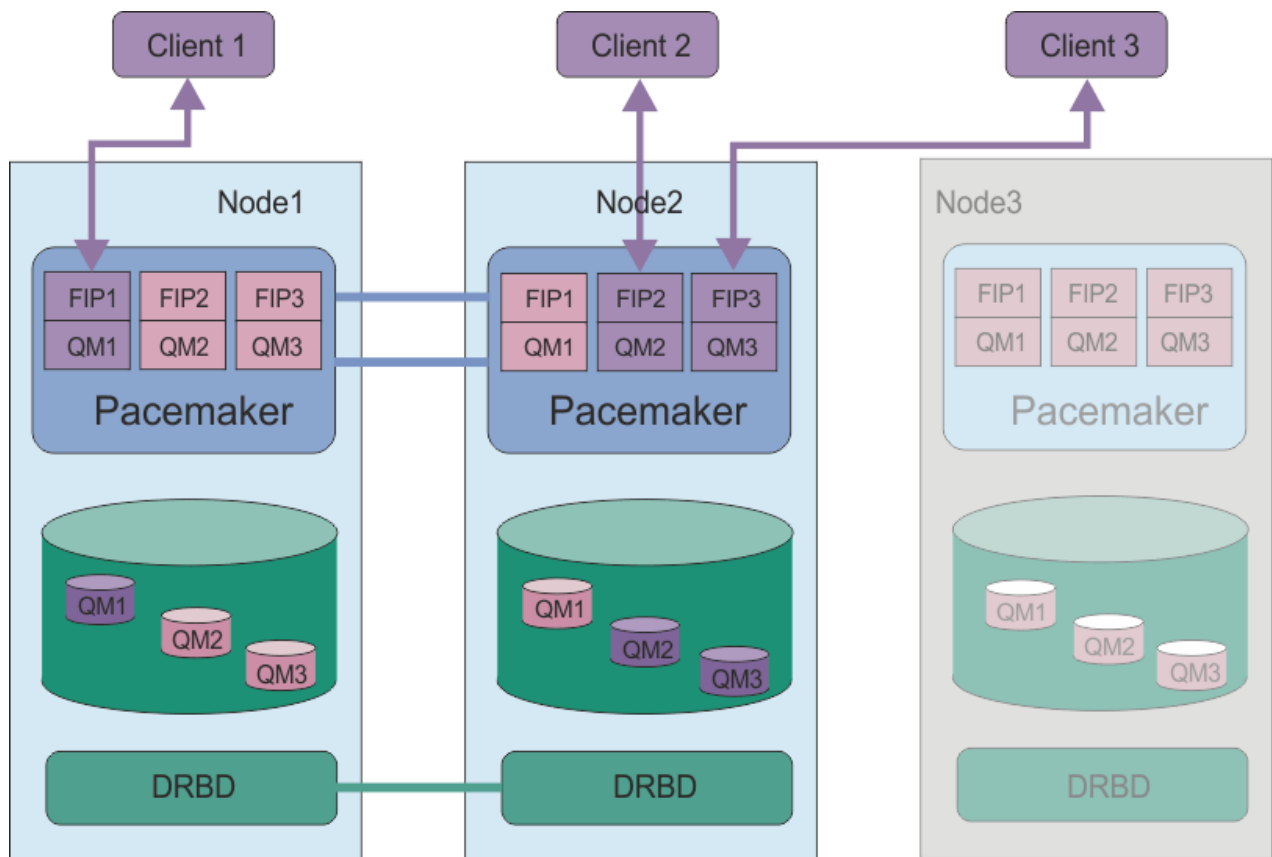


Figure 79. Example after node3 fails

Note: When queue managers fail over to another node they retain the state they had at failover. Queue managers that were running are started, queue managers that were stopped remain stopped.

Related tasks

[Installing RDQM \(replicated data queue managers\)](#)

[Applying maintenance level updates for RDQM](#)

[Migrating replicated data queue managers](#)

[Troubleshooting RDQM configurations](#)

Linux Requirements for RDQM HA solution

You must meet a number of requirements before you configure the RDQM high availability (HA) group.

System requirements

Before you configure the RDQM HA group, you must complete some configuration on each of the three servers that are to be part of the HA group.

- Each node requires a volume group named `drbdpool1`. The storage for each replicated data queue manager is allocated as a separate logical volume per queue manager from this volume group. For the best performance, this volume group should be made up of one or more physical volumes that correspond to internal disk drives (preferably SSDs). You can create `drbdpool1` before or after you have installed the RDQM HA solution, but you must create `drbdpool1` before you actually create any RDQMs. Check your volume group configuration by using the **vg**s command. The output should be similar to the following:

```
VG          #PV #LV #SN Attr   VSize  VFree
drbdpool1   1   9   0 wz--n- <16.00g <7.00g
rhe1        1   2   0 wz--n- <15.00g  0
```

In particular, check that there is no `c` character in the sixth column of the attributes field (`Attr`) (for example, `wz--nc`). The `c` indicates that clustering is enabled, and if it is you must delete the volume group and recreate it without clustering.

- After you have created the `drbdpool` volume group, do nothing else with it. IBM MQ manages the logical volumes created in `drbdpool`, and how and where they are mounted.
- Each node requires up to three interfaces that are used for configuring the RDQM support:
 - A primary interface for Pacemaker to monitor the HA group.
 - An alternate interface for Pacemaker to monitor the HA group.
 - An interface for the synchronous data replication, which is known as the replication interface. This should have sufficient bandwidth to support the replication requirements given the expected workload of all of the replicated data queue managers running in the HA group.

You can configure the HA group so that the same IP address is used for all three interfaces, a separate IP address is used for each interface, or the same IP address is used for primary and alternate and a separate IP address for the replication interface.

For maximum fault tolerance, these interfaces should be independent Network Interface Cards (NICs).

- DRBD requires that each node in the HA group has a valid internet host name (the value that is returned by `uname -n`), as defined by RFC 952 amended by RFC 1123.
- If there is a firewall between the nodes in the HA group, then the firewall must allow traffic between the nodes on a range of ports. A sample script is provided, `/opt/mqm/samp/rdqm/firewalld/configure.sh`, that opens up the necessary ports if you are running the standard firewall in RHEL. You must run the script as root. If you are using some other firewall, examine the service definitions `/usr/lib/firewalld/services/rdqm*` to see which ports need to be opened. The script adds the following permanent firewalld service rules for DRBD, Pacemaker, and IBM MQ:
 - `MQ_INSTALLATION_PATH/samp/rdqm/firewalld/services/rdqm-drbd.xml` allows TCP ports 7000-7100.
 - `MQ_INSTALLATION_PATH/samp/rdqm/firewalld/services/rdqm-pacemaker.xml` allows UDP ports 5404-5407
 - `MQ_INSTALLATION_PATH/samp/rdqm/firewalld/services/rdqm-mq.xml` allows TCP port 1414 (you must edit the script if you require a different port)
- If the system uses SELinux in Enforcing mode, you might need to run the following command:

```
semanage permissive -a drbd_t
```

If you have installed the `drbd-selinux` package, you do not need to run **semanage**. You must either have this package installed on every node, or run **semanage** on every node.

- IBM MQ RDQM environments support only EXT4 filesystems.

 RDQM uses ICMP ping packets to verify remote connections while configuring pacemaker and creating queue managers for RDQM DR and HA/DR configurations. If required, from IBM MQ 10.0.0, you can set the `AMQ_RDQM_NO_PING_CHECKS` environment variable to disable ICMP pings. By default, `AMQ_RDQM_NO_PING_CHECKS` is not honored for a non-root user with sudo privileges. Either set the environment variable and run the commands as root user, or update sudoers rules to include the environment variable. See [“Environment variables descriptions”](#) on page 65.

Network requirements

It is recommended that you locate the three nodes in the RDQM HA group in the same data center.

If you do choose to locate the nodes in different data centers, then be aware of the following limitations:

- Performance degrades rapidly with increasing latency between data centers. Although IBM will support a latency of up to 5 ms, you might find that your application performance cannot tolerate more than 1 to 2 ms of latency.

- The data sent across the replication link is not subject to any additional encryption beyond that which might be in place from using IBM MQ AMS.

You can optionally configure a floating IP address to enable a client to use the same IP address for a replicated data queue manager (RDQM) regardless of which node in the HA group it is running on. The floating address binds to a named physical interface on the primary node for the RDQM. If the RDQM fails over and a different node becomes the primary, the floating IP is bound to an interface of the same name on the new primary. The physical interfaces on the three nodes must all have the same name, and belong to the same subnet as the floating IP address.

User requirements for configuring the cluster

You can configure the RDQM HA group as user `root`. If you do not want to configure as `root`, you configure as a user in the `mqm` group instead. For a user in the `mqm` group to configure the RDQM cluster, you must meet the following requirements:

- The `mqm` user must be able to use `sudo` to run commands on each of the three servers that make up the RDQM HA group.
- If the `mqm` user can use SSH without a password to run commands on each of the three servers that make up the RDQM HA group, then the user needs to run commands on only one of the servers.
- The `mqm` user must have the same UID on all three servers.
- The `mqm` group must have the same GID on all three servers.

You must configure `sudo` so that the `mqm` user can run the following commands with root authority:

```
/opt/mqm/bin/crtmqm
/opt/mqm/bin/dltmqm
/opt/mqm/bin/xdqmadm
/opt/mqm/bin/xdqmstatus
```

User requirements for working with queue managers

To create, delete, or configure replicated data queue managers (RDQMs) you must use a user ID that belongs to both the `mqm` and `haclient` groups (the `haclient` group is created during installation of Pacemaker).

FIPS and TLS libraries

You can set the environment variable `GNUTLS_FORCE_FIPS_MODE` to 1 to force the TLS library into FIPS mode if FIPS 140-2 support is available. For more information about this environment variable, see https://gnutls.org/manual/html_node/FIPS140_002d2-mode.html.

Setting up passwordless SSH and sudo access

You can set up passwordless SSH and `sudo` access so that you only need issue configuration commands on one node in the HA group. (Setting up such access is optional, alternatively you can run commands on each node.)

About this task

To set up passwordless SSH you must configure the `mqm` id on each node, then generate a key on each node for that user. You then distribute the keys to the other nodes, and test the connection to add each node to the list of known hosts. Finally you lock down the `mqm` id, then create `sudo` access for the `mqm` id on each node.

The main advantage of passwordless SSH is that you can co-ordinate RDQM actions, such as the creation and deletion of the HA group and of HA Queue managers, from one node. Using passwordless SSH does require the set up described in this topic, and does have some security implications:

- Anyone who can gain access to the mqm user on one node, can SSH over to the same user on the other two nodes (this is mitigated by removing the password and locking the ID as suggested at the end of this procedure).
- There are general security principles around the protection of the private key.
- SSH communication is performed on the HA primary and alternative interfaces of the HA nodes.

The following RDQM commands are affected by the configuration of passwordless SSH, and are impacted by its absence:

- **crtmqm -sx**: without passwordless SSH, you have to run **crtmqm -sxs** on your two secondary nodes.
- **dlmqm**: without passwordless SSH, you have to issue the command on all HA nodes.
- **rdqmadm -c** or **rdqmadm -u**: without passwordless SSH, you have to run the command on all HA nodes.

You can remove the passwordless SSH set up and sudo access at the end of this procedure, if required. You can delete the mqm auth key on your nodes, or, if you want to be able to use the set up again, you can disable the public auth SSH.

Note: The instructions assume that you are defining an HA group with separate primary, alternate, and replication interfaces, and you therefore define passwordless SSH access over the primary and alternate interfaces. If you plan to configure a system with a single IP address, then you define passwordless SSH access over that single interface. If you plan to configure a system with two IP addresses for HA_Primary and HA_Replication, the ssh must be configured for the HA_Primary address.

You can then create sudo access for the mqm id on each node.

Procedure

1. To set up passwordless SSH:

- On each of the three nodes, complete the following steps to set up the mqm user and generate an SSH key:

- Change the mqm home directory to /home/mqm:

```
useimod -d /home/mqm mqm
```

- Create the /home/mqm directory:

```
mkhomedir_helper mqm
```

- Add the mqm password:

```
passwd mqm
```

- Run the interactive shell as mqm:

```
su mqm
```

- Generate the mqm authentication key:

```
ssh-keygen -t rsa -f /home/mqm/.ssh/id_rsa -N ''
```

- On each of the three nodes, complete the following steps to add that node's key to the other two nodes and test the connections for each nodes primary and (if used) alternate addresses:

- Add the key to the remote nodes:

```
ssh-copy-id -i /home/mqm/.ssh/id_rsa.pub remote_node1_primary_address
ssh-copy-id -i /home/mqm/.ssh/id_rsa.pub remote_node1_alternate_address
ssh-copy-id -i /home/mqm/.ssh/id_rsa.pub remote_node2_primary_address
ssh-copy-id -i /home/mqm/.ssh/id_rsa.pub remote_node2_alternate_address
```

- Check passwordless SSH and update known_hosts for remote nodes:


```
ssh remote_node1_primary_address uname -n
ssh remote_node1_alternate_address uname -n
ssh remote_node2_primary_address uname -n
ssh remote_node2_alternate_address uname -n
```

For each connection, you are prompted to confirm that you want to proceed. Confirm for each one to update the `known_hosts`. You must complete this before you attempt to configure the HA group using passwordless SSH.

iii) Exit the interactive shell as `mqm`:

```
exit
```

c) On each node, as root, complete the following steps to remove the `mqm` password and lock the id:

i) Remove the `mqm` password:

```
passwd -d mqm
```

ii) Lock `mqm`:

```
passwd -l mqm
```

2. On each node, as root, set up sudo access for the `mqm` user by creating the file `/etc/sudoers.d/mqm` containing the following text:

```
mqm ALL=(root) NOPASSWD: /opt/mqm/bin/crtmqm, /opt/mqm/bin/dltmqm, /opt/mqm/bin/
rdqmadm, /opt/mqm/bin/rdqmstatus, /opt/mqm/bin/rdqmdr
```

3. Optionally, when you have created your RDQM configuration, remove the passwordless SSH set up by reversing these steps as follows:

- a. Remove `/etc/sudoers.d/mqm` on all nodes.
- b. Unlock the `mqm` user (if desired).
- c. Remove the public/private keys from the nodes.
- d. Remove entry from `authorized_keys` file.

Performance capacity for high availability

You should consider the capacity of the RDQM high availability (HA) configuration when planning for performance.

As a general rule, you should plan on an absolute maximum of 30 to 40 queue managers in an RDQM HA configuration. Be aware that if you approach this top limit, then it can take five minutes or more to restore the availability of all queue managers in the event of a failover. Too many RDQM HA queue managers also affects the performance of non-RDQM HA queue managers.

If you have more than the recommended maximum number of queue managers, as well as excessive switchover times, you might observe unwanted failed actions that require manual intervention. Note the following points:

- The more RDQM HA queue managers you have, the more likely you are to experience failed actions.
- Failed actions are most likely to occur in RDQM HA switchover events, but can occur in steady state operations too.
- With very large numbers of RDQM HA queue managers, some failed actions might require assistance from IBM Support to recover.

Expect worse performance in terms of failover times and failed actions as the messaging load increases.

Once configured, you should test your system in both of the following conditions with your anticipated workload:

1. A planned switchover, where you suspend an RDQM HA queue manager.
2. An unplanned failure (for example, loss of power or loss of network). Loss of communication causes the RDQM HA queue managers to be started on another node.

Linux Defining the Pacemaker cluster (HA group)

The HA group is a Pacemaker cluster. You define the Pacemaker cluster by editing the `/var/mqm/rdqm.ini` file and running the **rdqmadm** command.

About this task

See <https://clusterlabs.org/pacemaker/> for information about Pacemaker. You can create the Pacemaker cluster as a user in the `mqm` group if the `mqm` user can use `sudo`. If the user can also SSH to each server without a password, then you only need edit the `rdqm.ini` file and run **rdqmadm** on one of the servers to create the Pacemaker cluster. Otherwise you must create the file and run the command as `root` on each of the servers that are to be nodes.

The `rdqm.ini` file gives the IP addresses that are used by RDQM for the nodes in the Pacemaker cluster. You must give the name of each node, which must be the hostname as returned by the **uname -n** command.

An RDQM HA group can be configured to use either one, two, or three IP addresses:

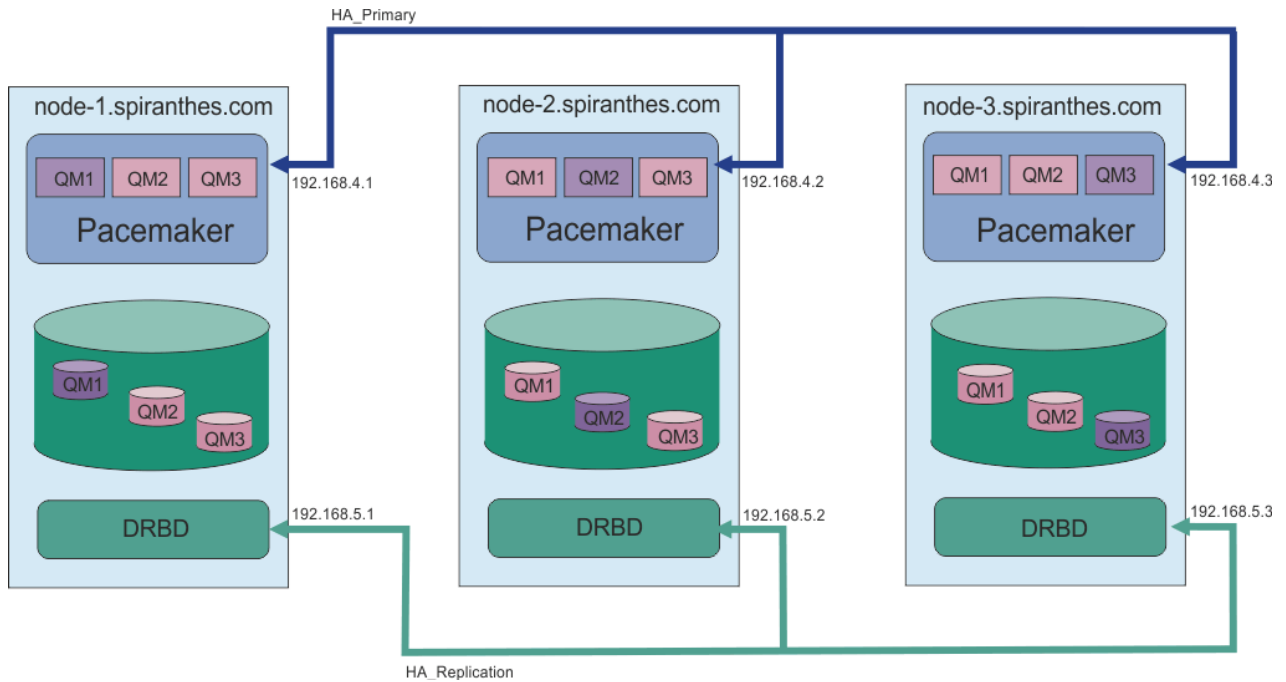
- One IP address: Heartbeats and replication share the same link
- Two IP addresses: Heartbeats and replication use separate links
- Three IP addresses: One link for replication, and two separate links for heartbeats

These options are provided to support different deployment patterns for RDQM. The different options can be used to maximize the resiliency of the RDQM solution based on the environment that is used. The configurations that use either two or three IP addresses are primarily intended for deployments where granular control is required over which physical network links the heartbeats and replication traffic use to configure redundancy for the connectivity between nodes. Alternatively, highly available and resilient connectivity can be implemented at the network layer, for example by using link aggregation. With link aggregation, multiple physical network links are used to provide a single logical link that can continue to function if individual physical links fail. If RDQM is deployed in an environment where the network connectivity is virtualized, and/or where resilient connectivity is implemented at the network layer, then using a single IP address for both heartbeats and replication is usually preferable.

The following example illustrates using two IP addresses. Your `rdqm.ini` file has an `HA_Primary` and an `HA_Replication` field for each node, but no `HA_Alternate` field:

```
Node:
  Name=rdqm-node-1.spiranthes.com
  HA_Primary=192.168.4.1
  HA_Replication=192.168.5.1
Node:
  Name=rdqm-node-2.spiranthes.com
  HA_Primary=192.168.4.2
  HA_Replication=192.168.5.2
Node:
  Name=rdqm-node-3.spiranthes.com
  HA_Primary=192.168.4.3
  HA_Replication=192.168.5.3
```

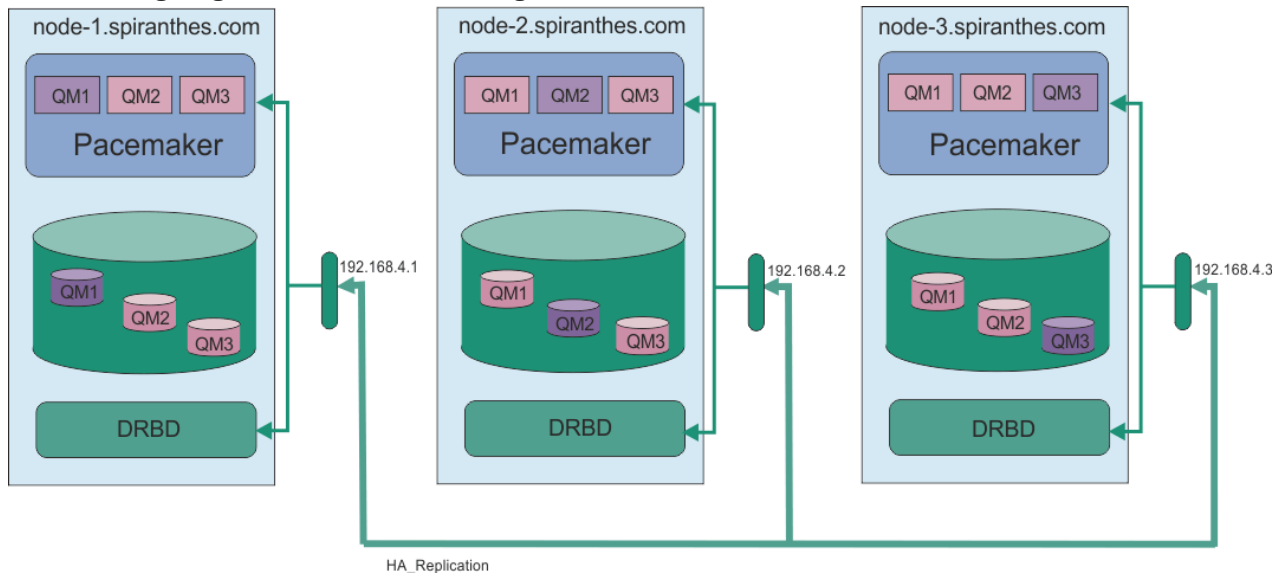
The following diagram illustrates this configuration:



The following example file shows the configuration for an example Pacemaker cluster that uses the HA_Replication interface for monitoring. In this case you only specify the HA_Replication interface:

```
Node:
  Name=rdqm-node-1.spiranthes.com
  HA_Replication=192.168.4.1
Node:
  Name=rdqm-node-2.spiranthes.com
  HA_Replication=192.168.4.2
Node:
  Name=rdqm-node-3.spiranthes.com
  HA_Replication=192.168.4.3
```

The following diagram illustrates this configuration:



The following example file shows the configuration for an example Pacemaker cluster that uses a separate IP address for each interface:

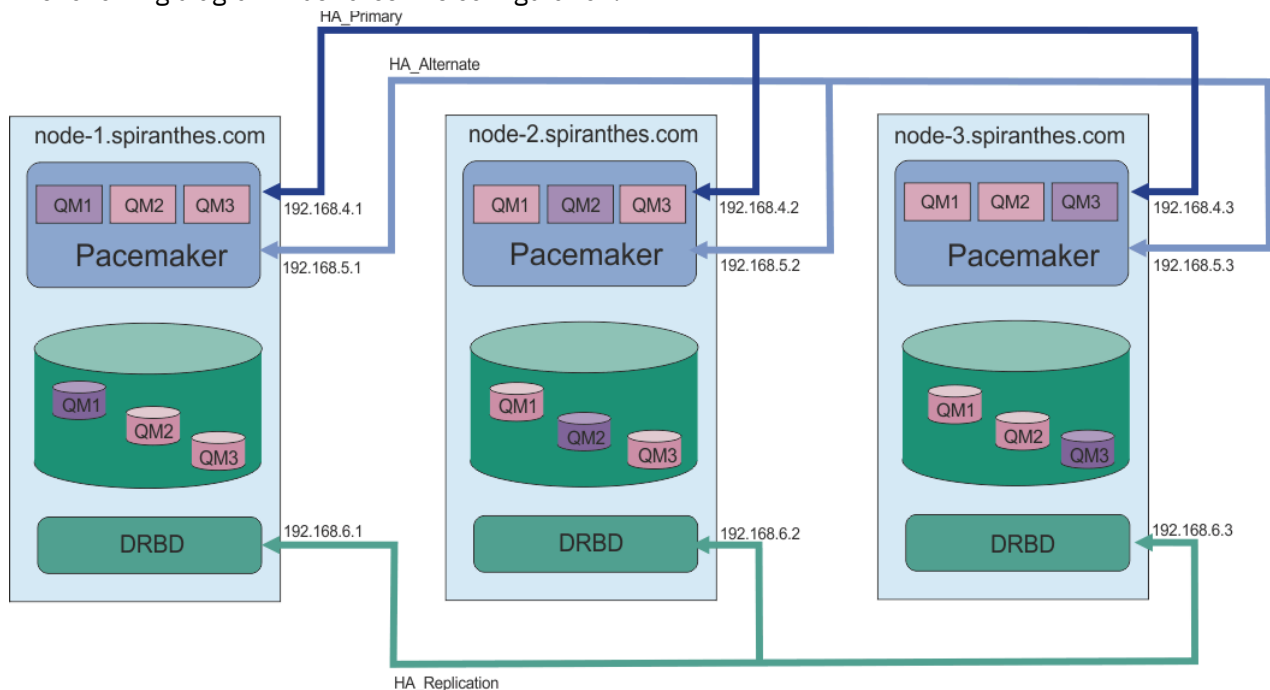
```
Node:
  Name=rdqm-node-1.spiranthes.com
```

```

HA_Primary=192.168.4.1
HA_Alternate=192.168.5.1
HA_Replication=192.168.6.1
Node:
Name=rdqm-node-2.spiranthes.com
HA_Primary=192.168.4.2
HA_Alternate=192.168.5.2
HA_Replication=192.168.6.2
Node:
Name=rdqm-node-3.spiranthes.com
HA_Primary=192.168.4.3
HA_Alternate=192.168.5.3
HA_Replication=192.168.6.3

```

The following diagram illustrates this configuration:



The order in which you specify the nodes must be the same in all the `rdqm.ini` files in your configuration. Your three nodes must have a common view as to which one is Node1, which one Node2 and so on.

Note: You can optionally specify that the heartbeat links in your group are encrypted, see [“Secure heartbeat for HA RDQM”](#) on page 637.

Procedure

- To define the Pacemaker cluster as user `root`:
 - a) Edit the `/var/mqm/rdqm.ini` file on one of the three servers so that the file defines the cluster.
 - b) Copy the file to the other two servers that will be nodes in the Pacemaker cluster.
 - c) Run the following command as `root` on each of the three servers:

```
rdqmadm -c
```

- To define the Pacemaker cluster as a user in the `mqm` group on each node:
 - a) Ensure that the user `mqm` can use `sudo` to run commands.
 - b) Edit the `/var/mqm/rdqm.ini` file on one of the three servers so that the file defines the Pacemaker cluster.
 - c) Copy the `/var/mqm/rdqm.ini` to the other two servers that will be nodes in the Pacemaker cluster.

d) Run the following command on each server:

```
rdqmadm -c
```

- To define the Pacemaker cluster as a user in the mqm group from one node:
 - a) Ensure that the user mqm can use **sudo** to run commands and can connect to each server using SSH without a password.
 - b) Edit the `/var/mqm/rdqm.ini` file on one of the three servers so that the file defines the Pacemaker cluster.
 - c) Run the following command:

```
rdqmadm -c
```

Related reference

[rdqmadm \(administer replicated data queue manager cluster\)](#)

Deleting the Pacemaker cluster (HA group)

The HA group is a Pacemaker cluster. You can delete a Pacemaker cluster configuration by running the **rdqmadm** command with the `-u` option.

About this task

You cannot delete the Pacemaker cluster configuration if any replicated data queue managers still exist on any of the nodes.

Procedure

- To delete the Pacemaker cluster configuration, enter the following command from any of the nodes:

```
rdqmadm -u
```

Related reference

[rdqmadm \(administer replicated data queue manager cluster\)](#)

Secure replication for HA RDQM

You can specify that replication links in your HA RDQM configuration are secured with TLS.

When you create a high availability (HA) queue manager in an RDQM configuration, you can specify that the links used to replicate data between the queue manager instances are secured with TLS.

To configure secure replication, complete the following steps:

1. Configure the required certificates.
2. Configure the `tlshd` service.
3. Specify the `-re` option when creating a new queue manager using **crtmqm**.

Complete the first two steps once per node. The certificates and `tlshd` configuration are shared by all RDQM queue managers on the node that are configured to use secure replication links.

The TLS protocols and cipher suites that can be used by the RDQM secure replication links are controlled by the active system-wide cryptographic policy. Use the Linux **update-crypto-policies** command to change the current active policy.

Configuring certificates

You must configure the certificates to be used by the replication links. The certificates that you create for each node can be used by any replication links. If you have more than one RDQM queue manager on a node that uses secure replication links, the same certificate is used.

Note: If any queue manager that runs on a node might ever be part of a DR/HA configuration, then a Subject Alternative Name (SAN) must be specified when the certificates are created. For more information, see [“Secure replication for DR/HA RDQM” on page 688](#).

To use secure replication links in a RDQM HA configuration you need three certificates, one certificate per node in the RDQM HA group.

To configure your certificates complete the following steps for each node in your RDQM configuration:

1. Use the following command to generate a private key:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -out key_file
```

where *key_file* is the name of the file that is created to store the private key.

2. Use the following command to create a certificate signing request:

```
openssl req -key key_file -subj "/CN=hostname" -new -out cert_req_file
```

where *key_file* is the file that contains the private key that you created in step “1” on page 634, *hostname* is the hostname of the node, and *cert_req_file* is the file that is created to store the certificate signing request.

Set the certificate Common Name (CN) to the hostname of the RDQM node.

You can also specify a CN that differs from the RDQM node hostname. If you do so, ensure that the certificate is created with a Subject Alternative Name (SAN) that contains the RDQM node hostname.

3. Send the certificate signing request to your certificate authority (CA) to be signed. When you receive the signed certificate from the CA, copy the private key, signed certificate, and CA certificate files to the directory where you want to store these files. For example, the `/etc/db2d.d/` directory.
4. Set the file permissions and ownership of the private key and certificate files to ensure that only the root user can view the private keys.
 - Change the ownership of the private key and certificate file to the root user.
 - Set the permissions on the private key file so that only the root user has read and write access.

The following example script creates an internal certificate authority and three certificates for use in a RDQM HA configuration.

Important: This example is not suitable for a production environment, and is intended only as an example to set up a test environment quickly. Certificate management is a complex subject for advanced users. For production, you must consider things like rotation, revocation, key length, and much more.

Run the following script on one of the nodes:

```
#!/bin/bash

# The CA name
ca_name="RDQM Test CA"

# The CA filename
ca_filename="rdqm_test_ca"

# Host certificates to generate
test_hosts=("alice" "bob" "charlie")

mkdir certificates && cd ./certificates

# Create private key for use with private CA using OpenSSL
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 \
-out $ca_filename.key

# Create a private CA certificate using OpenSSL
openssl req -key $ca_filename.key -new -x509 -days 3650 -subj "/CN=$ca_name" \
-out $ca_filename.crt

# Create certificates for each node
for host in ${test_hosts[@]}; do

    # Create a private key
```

```

openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 \
-out $host.key

# Generate a CSR
openssl req -key $host.key -subj "/CN=$host" -new -out $host.csr

# Create a valid certificate signed by the private CA
openssl x509 -req -in $host.csr -CA $ca_filename.crt \
-CAkey $ca_filename.key -CAcreateserial -days 365 -out $host.crt

# Remove sequential serial number
rm -f $ca_filename.srl

# Remove CSR
rm -f $host.csr

done

```

The example generates the following files in the certificates directory:

- `rdqm_test_ca.crt`
- `rdqm_test_ca.key`
- `alice.crt`
- `alice.key`
- `bob.crt`
- `bob.key`
- `charlie.crt`
- `charlie.key`

Deploy the certificates by completing the following steps:

1. Copy the CA certificate (`rdqm_test_ca.crt`) to each node's `/etc/drbd.d/` directory. (These are example paths, you can choose another location if required.)
2. Copy each node's certificate (`.crt` file) and private key (`.key` file) to `/etc/drbd.d/`. (These are example paths, you can choose another location if required.)
3. Change file permissions and ownership for the private keys and certificate on each node. This ensures that only the root user can view the contents of the private keys:

```

chmod 600 /etc/drbd.d/*.key
chown root:root /etc/drbd.d/*.key
chown root:root /etc/drbd.d/*.crt

```

4. Store the certificate authority's private key in a secure location.

Configuring the tlshd service

This `tlshd` service needs to be configured and then enabled and started, before any DRBD TLS handshakes can be completed. The `tlshd` service is provided in a package that is installed with the `drbd-utils-9` packages.

The configuration file for this service is `/etc/tlshd.conf` and the initial contents are:

```

[debug]
loglevel=0
tls=0
nl=0

[authenticate]
#keyrings= <keyring>;<keyring>;<keyring>

[authenticate.client]
#x509.truststore= <pathname>
#x509.certificate= <pathname>
#x509.private_key= <pathname>

[authenticate.server]
#x509.truststore= <pathname>

```

```
#x509.certificate= <pathname>
#x509.private_key= <pathname>
```

Before enabling and starting the `tlshd` service, you must specify values for the following properties in both the `authenticate.client` and `authenticate.server` sections (DRBD can establish its connections in either direction, unless there is a firewall blocking one direction):

x509.truststore

The path to the file that contains the certificate of the CA that signed the certificates used by the RDQM nodes. If a complex trust chain is required, it must be captured in a single file.

x509.certificate

The path to the file that contains the CA-signed certificate for the RDQM node.

x509.private_key

The path to the file that contains the private key for the RDQM node. The file must be owned by root and read/write permissions must be granted to only the root user.

The paths that are specified in these properties must all identify local files.

You can also optionally specify a certificate revocation list (CRL). You must set this in both the `authenticate.client` and `authenticate.server` sections:

x509.crl

The path to a file containing PEM-encoded certificate revocation lists (CRL) that are to be used to verify the revocation status of certificates during each handshake. If this option is not specified, CRL checking is skipped.

If you created the certificates using the example script in [“Configuring certificates”](#) on page 633, then you would configure the file as follows for server `alice`:

```
[authenticate.client]
x509.truststore=/etc/drbd.d/rdqm_test_ca.crt
x509.certificate=/etc/drbd.d/alice.crt
x509.private_key=/etc/drbd.d/alice.key

[authenticate.server]
x509.truststore=/etc/drbd.d/rdqm_test_ca.crt
x509.certificate=/etc/drbd.d/alice.crt
x509.private_key=/etc/drbd.d/alice.key
```

Before you enable the service for the first time, enter the following command:

```
systemctl daemon-reload
```

To start the `tlshd` service, enter the following command:

```
systemctl enable --now tlshd
```

Creating the HA RDQM

You create an HA queue manager with secure replication links by using the **`crtmqm`** command with the `-re` option, for example:

```
crtmqm -sx -re HAqm
```

The command attempts to use SSH to connect to the other nodes in the cluster as the `mqm` user. If connection is successful, the secondary instances of the queue manager are created on the other nodes, and these secondary instances also use secure replication links.

Use the **`rdqmstatus`** command to confirm that the replication links are secured with TLS. For more information, see [“Viewing HA RDQM and HA group status”](#) on page 650.

For more information about creating an HA RDQM, see [“Creating an HA RDQM”](#) on page 637.

Secure heartbeat for HA RDQM

You can specify that the heartbeat links in your HA RDQM configuration are encrypted.

When you create a high availability (HA) group in an RDQM configuration, you can specify that the links used for the heartbeats that monitor the availability of your nodes are encrypted.

To configure secure replication, complete the following steps:

1. Create a shared secret and distribute it to all the nodes.
2. Specify the `-c` `-e` options when creating a new RDQM HA group using **rdqmadm**.

When you enable encryption on the heartbeat links, the following settings are implemented:

- The option `crypto_cipher` is set to `aes256`.
- The option `crypto_hash` is set to `sha512`.

These values cannot be configured, the only other valid value is `none` when encryption is not enabled.

For more details about the crypto options that Pacemaker/Corosync/Knet support, see the Pacemaker documentation, <https://clusterlabs.org/pacemaker/>.

Configuring a shared secret

You create the shared secret by running the **corosync-keygen** command as root on one of your three nodes. This command creates a file named `/etc/corosync/authkey` with the following ownership and permissions:

```
-r-----. 1 root root 256 Jul 22 13:29 authkey
```

You must copy this file to the same location on the other two nodes, with the same ownership and permissions, before you create the RDQM HA group by using the **rdqmadm** command.

Creating the RDQM HA group

You create an RDQM HA group by following the instructions in “[Defining the Pacemaker cluster \(HA group\)](#)” on page 630. This topic guides you through the options available in setting up your group.

When you come to create the group, you use the **rdqmadm** command, and you can at this point specify an additional `-e` option to specify that encryption is enabled on the heartbeat links:

```
rdqmadm -c -e
```

You must ensure that this command runs on all three nodes in your group if passwordless ssh is not enabled.

If you want to subsequently disable encryption for this group, you need to delete and then recreate the group without the `-e` option.

Creating an HA RDQM

You use the **crtmqm** command to create a high availability replicated data queue manager (RDQM).

About this task

You can create a high availability replicated data queue manager (RDQM) as a user in the `mqm` group if the `mqm` user can use `sudo`, otherwise you must be `root` to create an RDQM. If the `mqm` user can SSH to each node without a password, then you only need run the create RDQM command on one node to create the RDQM on all three nodes, otherwise you must run commands on all three nodes.

Note: For information about the number of queue managers that can be created in a single RDQM HA group, see “[Performance capacity for high availability](#)” on page 629.

The following points give some guidance about sizing the queue manager file system:

1. When creating an RDQM queue manager, a file system is allocated to store queue manager data and logs. It is important to size this file system appropriately so that the queue manager can record ongoing activity in its logs, and store application messages on queues. When sizing the file system, consider peak messaging requirements, future workload growth, and application outages that might cause messages to build up on queues. For guidance on calculating the size of the queue manager recovery log, see [“How large should I make my log filesystem?”](#) on page 737. When calculating the storage requirements for application messages, the size and number of messages needs to be considered, plus their MQMD header and any message properties they have.
2. RDQM queue manager file systems cannot be dynamically resized. You must back up then restore an RDQM queue manager with a larger file system if this is required, see [“Resizing the filesystem for an HA RDQM queue manager”](#) on page 643.
3. You can limit the size of individual queues on disk by using local queue attributes, such as MAXDEPTH and MAXFSIZE. See [Modifying IBM MQ queue files](#).
4. You should monitor your ongoing disk usage and respond appropriately if disk usage increases before file system usage becomes critical. File system usage can be monitored by using either platform/operating system capabilities, or by subscribing to metrics published to IBM MQ system topics that are described at [Metrics published on the system topics](#).

V 10.0.0 You can, if required, specify that the replication links between the instances of the HA RDQM are secured with TLS. To use secure replication links, you must configure the certificates and tlshd service before the HA RDQM is created. For more information, see [“Secure replication for HA RDQM”](#) on page 633.

Procedure

- To create an RDQM as a user in the mqm group:
 - a) Ensure that the mqm user can use **sudo** to run commands and can connect to each server using SSH without a password.
 - b) Enter the following command:

```
crtmqm -sx [-re] [-fs FilesystemSize] qmname
```

where *qmname* is the name of the replicated data queue manager. You can optionally specify the file system size for the queue manager (that is, the size of the logical volume which is created in the drbdpool volume group). V 10.0.0 You can also optionally specify the **-re** option to secure the replication links between the RDQM instances with TLS.

The command attempts to use SSH to connect to the other nodes in the cluster as the mqm user. If connection is successful, the secondary instances of the queue manager are created on the nodes. Otherwise, you must create the secondary instances and then run the **crtmqm -sx** command (as described for user root).

- To create an RDQM as user root:
 - a) Enter the following command on each of the nodes that are to host secondary instances of the RDQM:

```
crtmqm -sxs [-re] [-fs FilesystemSize] qmname
```

where *qmname* is the name of the replicated data queue manager. You can optionally specify the file system size for the queue manager (that is, the size of the logical volume which is created in the drbdpool volume group). You must specify the same file system size for the RDQM on all three nodes in the HA group. The size is a numeric value, which is specified in GB. You can specify a value in MB by entering the value followed by the character M. V 10.0.0 You can also optionally specify the **-re** option to secure the replication links between the RDQM instances with TLS.

The command creates a secondary instance of the RDQM.

- b) On the remaining node, enter the following command:

```
crtmqm -sx [-re] [-fs FilesystemSize] qmname
```

where *qmname* is the name of the replicated data queue manager. You can optionally specify the file system size for the queue manager. The size is a numeric value, which is specified in GB. You can specify a value in MB by entering the value followed by the character M. [V.10.0.0](#) You can also optionally specify the `-re` option to secure the replication links between the RDQM instances with TLS.

The command determines if the secondary instance of the queue manager exist on the other two nodes. If secondaries exist, the command creates and starts the primary queue manager. If the secondaries do not exist, you are instructed to run the **crtmqm -sxs** command on each of the nodes.

Apart from the DataPath (**-md**) and LogPath (**-ld**) arguments, all arguments that are valid for creating a standard Linux queue manager are also valid for a primary replicated data queue manager.

For more information about the **crtmqm** command, see [crtmqm \(create queue manager\)](#).

Note: When you create an RDQM, the next free port number above 7000 is allocated for the replication link. If it is discovered that the chosen port is used by another application, the **crtmqm** command fails with the AMQ6543 error and that port is added to an exclude list. You must delete the secondary instances of the queue manager, then run the **crtmqm** command again.

Related reference

[crtmqm](#)

Deleting an HA RDQM

You use the **dltmqm** command to delete a high availability replicated data queue manager (RDQM).

About this task

You must run the command to delete the RDQM on the RDQM's primary node. The RDQM must be ended first. You can run the command as an mqm user if that user has the necessary sudo privileges. Otherwise, you must run the command as root. After the resources associated with the primary queue manager have been deleted, the command attempts to delete the secondary queue managers using ssh to connect to the other nodes. If this deletion fails, you must run **dltmqm** manually on the other nodes to complete the process. On a secondary node, the command fails if the primary queue manager has not already been deleted.

Procedure

- To delete an RDQM, enter the following command:

```
dltmqm RDQM_name
```

Related reference

[dltmqm](#)

Migrating a queue manager to become an HA RDQM queue manager

You can migrate an existing queue manager to become a high availability (HA) replicated data queue manager (RDQM) by backing up its persistent data, then restoring the data to a newly created RDQM queue manager that has the same name.

About this task

HA Replicated Data Queue Managers require a dedicated logical volume (filesystem) and the configuration of disk replication and HA control. These components are only configured when a new queue manager is created. An existing queue manager can be migrated to use RDQM by backing up its persistent data,

then restoring the data to a newly created RDQM queue manager that has the same name. This procedure preserves the queue manager configuration, state, and persistent messages at the time the backup was created.

Note: You can only migrate a queue manager from a version of IBM MQ that is the same as or lower than the version where RDQM is installed. The operating system and architecture must also be the same. Otherwise, you must create a new queue manager on your target platform, see [Moving a queue manager to a different operation system](#).

You should satisfy the following conditions before you migrate a queue manager:

- Evaluate your high availability requirements and see [“RDQM high availability”](#) on page 622.
- Review the applications and queue managers that connect to the queue manager. Consider the changes required to route the connections to the RDQM node where the queue manager is running. For example, if you configure RDQM high availability you might consider using a floating IP address, see [“Creating and deleting a floating IP address”](#) on page 646.
- Provision, or identify existing, RDQM nodes for your chosen configuration. For information about the system requirements for RDQM see [“Requirements for RDQM HA solution”](#) on page 625.
- Install IBM MQ Advanced, which includes the RDQM feature, on each node.
- Configure the RDQM HA group configuration, see [“Defining the Pacemaker cluster \(HA group\)”](#) on page 630.
- Optionally, verify the RDQM configuration using a test queue manager, which can then be deleted. Testing the configuration is recommended to identify and resolve any problems before migrating the queue manager.
- Review the security configuration for the queue manager, then replicate the required local users and groups on each RDQM node.
- Review the queue manager and channel configuration to determine if API exits, channel exits, or data conversion exits are used. Install the required exits on each RDQM node.
- Review any queue manager services that have been defined, then install and configure the required processes on each RDQM node.

Procedure

1. Back up the existing queue manager:

- a) Stop the existing queue manager by issuing a wait shutdown command `endmqm -w`, or an immediate shutdown command `endmqm -i`. This step is important to ensure the data in the backup is consistent.
- b) Determine the location of the queue manager data directory by viewing the IBM MQ configuration file, `mqm.ini`. On Linux, this file is located in the `/var/mqm` directory. For more information about `mqm.ini`, see [“IBM MQ configuration file, mqm.ini”](#) on page 96.

Locate the `QueueManager` stanza for the queue manager in the file. If the stanza contains a key named `DataPath` then its value is the queue manager data directory. If the key does not exist, then the queue manager data directory can be determined using the values of the `Prefix` and `Directory` keys. The queue manager data directory is a concatenation of these values, of the form `prefix/qmgrs/directory`. For more information about the `QueueManager` stanza, see [“QueueManager stanza of the mqm.ini file”](#) on page 106.

- c) Create a backup of the queue manager data directory. On Linux, you can do this by using the **tar** command. For example, to back up the data directory for a queue manager you can use the following command. Note the last parameter of the command, which is a single period (dot):

```
tar -cvzf qm-data.tar.gz -C queue_manager_data_dir .
```

- d) Determine the location of the queue manager log directory by viewing the IBM MQ queue manager configuration file `qm.ini`. This file is located in the queue manager data directory. For more information about the file, see [“Queue manager configuration files, qm.ini”](#) on page 108.

The queue manager log directory is defined as the value of the LogPath key in the Log stanza. For information about the stanza, see [“Log stanza of the qm.ini file” on page 142](#).

- e) Create a backup of the queue manager log directory. On Linux, you can do this by using the tar command. For example, to back up the log directory for a queue manager you can use the following command. Note the last parameter of the command, which is a single period (dot):

```
tar -cvzf qm-log.tar.gz -C queue_manager_log_dir .
```

- f) Create a backup of any certificate repositories used by the queue manager if they are not located in the queue manager data directory. Ensure that both the key database file and the password stash file are backed up. For information about the queue manager key repository, see [The SSL/TLS key repository](#) and [Locating the key repository for a queue manager](#). For information about locating the AMS key store if the queue manager is configured to use AMS Message Channel Agent (MCA) interception, see [Message Channel Agent \(MCA\) interception](#).
- g) The existing queue manager is no longer required, so it can be deleted. However, where possible, you should only delete the existing queue manager after it has been successfully restored on the target system. Deferring deletion ensures that the queue manager can be restarted if the migration process does not complete successfully.

Note: If you defer deletion of the existing queue manager, do not restart it. It is important the queue manager remains ended because further changes to its configuration or state are lost during the migration.

2. Prepare the primary RDQM node:

- a) Create a new RDQM queue manager with the same name as the queue manager that you backed up. Ensure the filesystem allocated for the RDQM queue manager by `crtmqm` is big enough to contain the data, primary logs and secondary logs for the existing queue manager, plus some additional space for future expansion. For information about how to create an RDQM queue manager, see [“Creating an HA RDQM” on page 637](#).
- b) Determine the primary RDQM node for the queue manager. For information about how to determine the primary node, see [rdqmstatus \(display RDQM status\)](#).
- c) On the primary RDQM node, if the RDQM queue manager is started, stop it by using the `endmqm -w` or `endmqm -i` command.
- d) On the primary RDQM node, determine the location of the data and log directories for the RDQM queue manager (use the methods described in steps 1b and 1d).
- e) On the primary RDQM node, delete the contents of the RDQM queue manager data and log directories, but not the directories themselves.

3. Restore the queue manager on the primary RDQM node:

- a) Copy the backups of the queue manager data and log directories to the primary RDQM node, plus any separate backups of certificate repositories used by the queue manager.
- b) Restore the backup of the queue manager data directory to the empty data directory for the new RDQM queue manager, ensuring that file ownership and permissions are preserved. If the backup was created using the example tar command in step 1c then the following command can be used by the root user to restore it:

```
tar -xvzpf qm-data.tar.gz -C queue_manager_data_dir
```

- c) Restore the backup of the queue manager log directory to the empty log directory for the new RDQM queue manager, ensuring that file ownership and permissions are preserved. If the backup was created using the example tar command in step 1e then the following command can be used by the root user to restore it:

```
tar -xvzpf qm-log.tar.gz -C queue_manager_log_dir
```

- d) Edit the restored queue manager configuration file, `qm.ini`, in the data directory for the RDQM queue manager. Update the value of the LogPath key in the Log stanza to specify the log directory for the RDQM queue manager.

Review other file paths that are defined in the configuration file and update them if necessary. For example, you might need to update the following paths:

- The path for error log files that are generated by diagnostic message services.
 - The path for exits that are required by the queue manager.
 - The path for switch load files if the queue manager is an XA transaction coordinator.
- e) If the queue manager is configured to use AMS Message Channel Agent (MCA) interception, copy the AMS key store to the new RDQM installation, then review and update the configuration. The key store must be available on each RDQM node, so if it is not located in the replicated filesystem for the queue manager it must be copied to each node instead. For more information, see [Message Channel Agent \(MCA\) interception](#).
- f) Verify that the queue manager is displayed by the **dspmq** command and its status is reported as ended. The following example shows sample output for an RDQM HA queue manager:

```
$ dspmq -o status -o ha
QMNAME(QM1) STATUS(Ended normally) HA(Replicated)
```

- g) Verify that the restored queue manager data has been replicated to the secondary RDQM nodes by using the **rdqmstatus** command to display the status for the queue manager. The HA status should be reported as Normal on each node. The following example shows sample output for an RDQM HA queue manager:

```
$ rdqmstatus -m QM1
Node:                mqhavam10-adm
Queue manager status: Ended normally
Queue manager file system: 50MB used, 0.2GB allocated [42%]
HA role:              Primary
HA status:             Normal
HA control:            Disabled
HA current location:   This node
HA preferred location: This node
HA floating IP interface: None
HA floating IP address: None

Node:                mqhavam11-adm
HA status:            Normal

Node:                mqhavam12-adm
HA status:            Normal
```

- h) Start the queue manager on the primary RDQM node.
- i) Connect to the queue manager and update the value of the SSLKEYR queue manager attribute to specify the new location of the queue manager certificate repository. By default, the value of this attribute is set to *queue_manager_data_directory/ssl/key*. The certificate repository must be located in the same location on each RDQM node. If the repository is not located in the replicated filesystem for the queue manager, then it must be copied to each node instead.
- j) Review the IBM MQ object definitions for the queue manager and update the value of object attributes that reference changed network settings, the IBM MQ installation directory, or the queue manager data directory, including the following objects:
- Local IP addresses used by listeners (IPADDR attribute).
 - Local IP addresses used by channels (LOCLADDR attribute).
 - Local IP addresses defined for cluster-receiver channels (CONNAME attribute).
 - Local IP addresses defined for communication information objects (GRPADDR attribute).
 - System paths defined for process and service object definitions.
- k) Stop and restart the queue manager to ensure the changes become effective.
- l) Repeat step 3j for remote queue managers, plus equivalent settings for applications, that connect to the migrated queue manager, including:
- Channel connection names (CONNAME attribute).

- Channel authentication rules that restrict inbound connections from the queue manager based on its IP address or hostname.
 - Client channel definition tables (CCDTs), domain name settings (DNS), network routing, or equivalent connection information.
- m) Perform a managed failover of the queue manager to each RDQM node to ensure the required configuration has been established successfully, see [“Setting the Preferred Location for an RDQM” on page 645](#).

Resizing the filesystem for an HA RDQM queue manager

To resize the filesystem for an existing high availability (HA) replicated data queue manager (RDQM) you back up its persistent data, then restore the data to a newly created RDQM queue manager that has the same name but a filesystem of a different size.

About this task

HA replicated data queue managers require a dedicated logical volume (filesystem) and the configuration of disk replication and HA control. These components are only configured when a new queue manager is created. The filesystem cannot be resized after it has been created because it must have the same size on each node. To resize the filesystem for an existing replicated data queue manager (RDQM) you can back up its persistent data, then restore the data to a newly created RDQM queue manager that has the same name but a filesystem of a different size. This procedure preserves the queue manager configuration, state, and persistent messages at the time the backup was created.

Procedure

1. Back up the existing RDQM queue manager on the primary RDQM node:
 - a) Determine the primary RDQM node for the queue manager. For information about how to determine the primary node, see [rdqmstatus \(display RDQM status\)](#).
 - b) On the primary RDQM node, if the RDQM queue manager is started, stop it by using the **endmqm -w** or **endmqm -i** command.
 - c) Determine the location of the queue manager data directory by viewing the IBM MQ configuration file, `mqs.ini`. On Linux, this file is located in the `/var/mqm` directory. For more information about `mqs.ini`, see [“IBM MQ configuration file, mqs.ini” on page 96](#).

Locate the `QueueManager` stanza for the queue manager in the file. The queue manager data directory is the value of the key named `DataPath`. For more information about the `QueueManager` stanza, see [“QueueManager stanza of the mqs.ini file” on page 106](#).

- d) Create a backup of the queue manager data directory. On Linux, you can do this by using the **tar** command. For example, to back up the data directory for a queue manager you can use the following command. Note the last parameter of the command, which is a single period (.) character:

```
tar -cvzf qm-data.tar.gz -C queue_manager_data_dir .
```

- e) Determine the location of the queue manager log directory by viewing the IBM MQ queue manager configuration file `qm.ini`. This file is located in the queue manager data directory. For more information about the file, see [“Queue manager configuration files, qm.ini” on page 108](#).

The queue manager log directory is defined as the value of the `LogPath` key in the `Log` stanza. For information about the stanza, see [“Log stanza of the qm.ini file” on page 142](#).

- f) Create a backup of the queue manager log directory. On Linux, you can do this by using the **tar** command. For example, to back up the log directory for a queue manager you can use the following command. Note the last parameter of the command, which is a single period (.) character:

```
tar -cvzf qm-log.tar.gz -C queue_manager_log_dir .
```

- g) Delete the existing RDQM queue manager.

2. Restore the queue manager with a filesystem of the required size:

- a) Create a new RDQM queue manager with the same name as the queue manager that you backed up. Ensure the filesystem allocated for the RDQM queue manager by **crtmqm** is the size you require, and it is big enough to contain the data, primary logs, and secondary logs for the existing queue manager, plus some additional space for future expansion. For information about how to create an RDQM queue manager, see [“Creating an HA RDQM” on page 637](#).
- b) Determine the primary RDQM node for the queue manager. For information about how to determine the primary node, see [rdqmstatus \(display RDQM status\)](#).
- c) On the primary RDQM node, if the RDQM queue manager is started, stop it by using the **endmqm -w** or **endmqm -i** command.
- d) On the primary RDQM node, determine the new location of the data and log directories for the RDQM queue manager (use the methods described in steps 1c and 1e).
- e) On the primary RDQM node, delete the contents of the RDQM queue manager data and log directories, but not the directories themselves.
- f) On the primary RDQM node, restore the backup of the queue manager data directory to the empty data directory for the new RDQM queue manager, ensuring that file ownership and permissions are preserved. If the backup was created using the example **tar** command in step 1d then the following command can be used by the root user to restore it:

```
tar -xvzpf qm-data.tar.gz -C queue_manager_data_dir
```

- g) On the primary RDQM node, restore the backup of the queue manager log directory to the empty log directory for the new RDQM queue manager, ensuring that file ownership and permissions are preserved. If the backup was created using the example **tar** command in step 1f then the following command can be used by the root user to restore it:

```
tar -xvzpf qm-log.tar.gz -C queue_manager_log_dir
```

- h) On the primary RDQM node, edit the restored queue manager configuration file, **qm.ini**, in the data directory for the new RDQM queue manager. Update the value of the LogPath key in the Log stanza to specify the log directory for the new RDQM queue manager that you determined in step 2d. Review other file paths that are defined in the configuration file and update them if necessary. For example, you might need to update the following paths:

- The path for error log files that are generated by diagnostic message services.
- The path for exits that are required by the queue manager.
- The path for switch load files if the queue manager is an XA transaction coordinator.

- i) Verify that the queue manager is displayed by the **dspmq** command and its status is reported as ended. The following example shows sample output for an RDQM HA queue manager:

```
$ dspmq -o status -o ha
QMNAME(QM1) STATUS(Ended normally) HA(Replicated)
```

- j) Verify that the restored queue manager data has been replicated to the secondary RDQM nodes by using the **rdqmstatus** command to display the status for the queue manager. The HA status should be reported as Normal on each node. The following example shows sample output for an RDQM HA queue manager:

```
$ rdqmstatus -m QM1
Node: mqhvm10-adm
Queue manager status: Ended normally
Queue manager file system: 50MB used, 0.2GB
allocated [42%]
HA role: Primary
HA status: Normal
HA control: Disabled
HA current location: This node
HA preferred location: This node
HA floating IP interface: None
HA floating IP address: None
Node: mqhvm11-adm
```


HA status:	Normal
Node:	mqhavam12-adm
HA status:	Normal

- k) Start the queue manager on the primary RDQM node.
- l) Perform a managed failover of the queue manager to each RDQM node to ensure the required configuration has been established successfully, see [“Setting the Preferred Location for an RDQM” on page 645](#).

Storing persistent application status

You can store persistent status information relating to applications along with other queue manager data.

Each IBM MQ queue manager has a dedicated filesystem for its persistent state, which includes both its queue data and recovery log. In an RDQM configuration the filesystem is backed by a logical volume that is replicated between the Linux systems (nodes). The filesystem includes a `userdata` directory that you can use to store persistent status information for your applications. So, when a replicated data queue manager moves to run on another node in your RDQM configuration, you have application context available as well as queue manager context. See [Directory content on Unix and Linux Systems](#).

If you choose to store application state in the `userdata` directory, you must be aware that data written to this location might consume the available disk space allocated to the queue manager. You must ensure that sufficient disk space remains available for the queue manager to write queue data, logs, and other persistent state information.

The `userdata` directory has `mqm` user and group ownership, and it is world readable so users can access it without needing to be in the IBM MQ administrator group (that is, `mqm`). You cannot modify the permissions of the `userdata` directory, but you can create content in it with whatever ownership and permissions you require.

During an RDQM queue manager failover the queue manager is ended, and its filesystem is unmounted on its current RDQM node. The filesystem is then mounted and the queue manager restarted on another node in the RDQM configuration. A filesystem cannot be unmounted if a process has an open handle for one of its files. To ensure a queue manager failover can complete, if the queue manager filesystem cannot be unmounted then processes that have an open file handle are sent a `SIGTERM` signal, followed by a `SIGKILL` if the open handles are not released. Your applications must be designed to respond correctly to `SIGTERM`. If applications or processes are configured as a queue manager service then, during a managed failover, they can be ended during queue manager shutdown before the filesystem is unmounted. If an application or process is not configured as a queue manager service, or an unmanaged failover occurs, such as a loss of quorum, then it is likely that signals will be sent to release the filesystem.

Setting the Preferred Location for an RDQM

The Preferred Location for a replicated data queue manager (RDQM) identifies the node where the RDQM should run if that node is available.

About this task

The Preferred Location is the name of the node on which Pacemaker should run the queue manager when the HA group is in a normal state (all nodes and connections available). The Preferred Location is initialized to the name of the primary node when the queue manager is created. You can run the commands to set the Preferred Location on any of the three nodes. You must be a user who belongs to both the `mqm` and `haclient` groups.

Procedure

- To assign the local or specified node as the Preferred Location for the named queue manager, enter the following command:

```
rdqmadm -p -m qmname [ -n nodename[,nodename] ]
```

where *qmname* is the name of the RDQM you are specifying the preferred location for, and *nodename* is optionally the name of the preferred node.

If the HA group is in a normal state and the Preferred Location is not the current primary node, the queue manager is stopped and restarted on the new Preferred Location. You can specify a comma-separated list of two node names to assign a second preference of Preferred Location.

- To clear the Preferred Location so that the queue manager does not automatically return to a node when it is restored, enter the following command:

```
rdqmadm -p -m qmname -d
```

Related reference

[rdqmadm \(administer replicated data queue manager cluster\)](#)

Creating and deleting a floating IP address

A floating IP address enables a client to use the same IP address for a replicated data queue manager (RDQM) regardless of which node in the HA group it is running on. (The use of a floating IP address is optional.)

About this task

You can create or delete a floating IP address by using the **rdqmint** command. The floating address binds to a named physical interface on the primary node for the RDQM. If the RDQM fails over and a different node becomes the primary, the floating IP is bound to an interface of the same name on the new primary. The physical interfaces on the three nodes must belong to the same subnet as the floating IP address. The following diagram illustrates the use of a floating IP address.

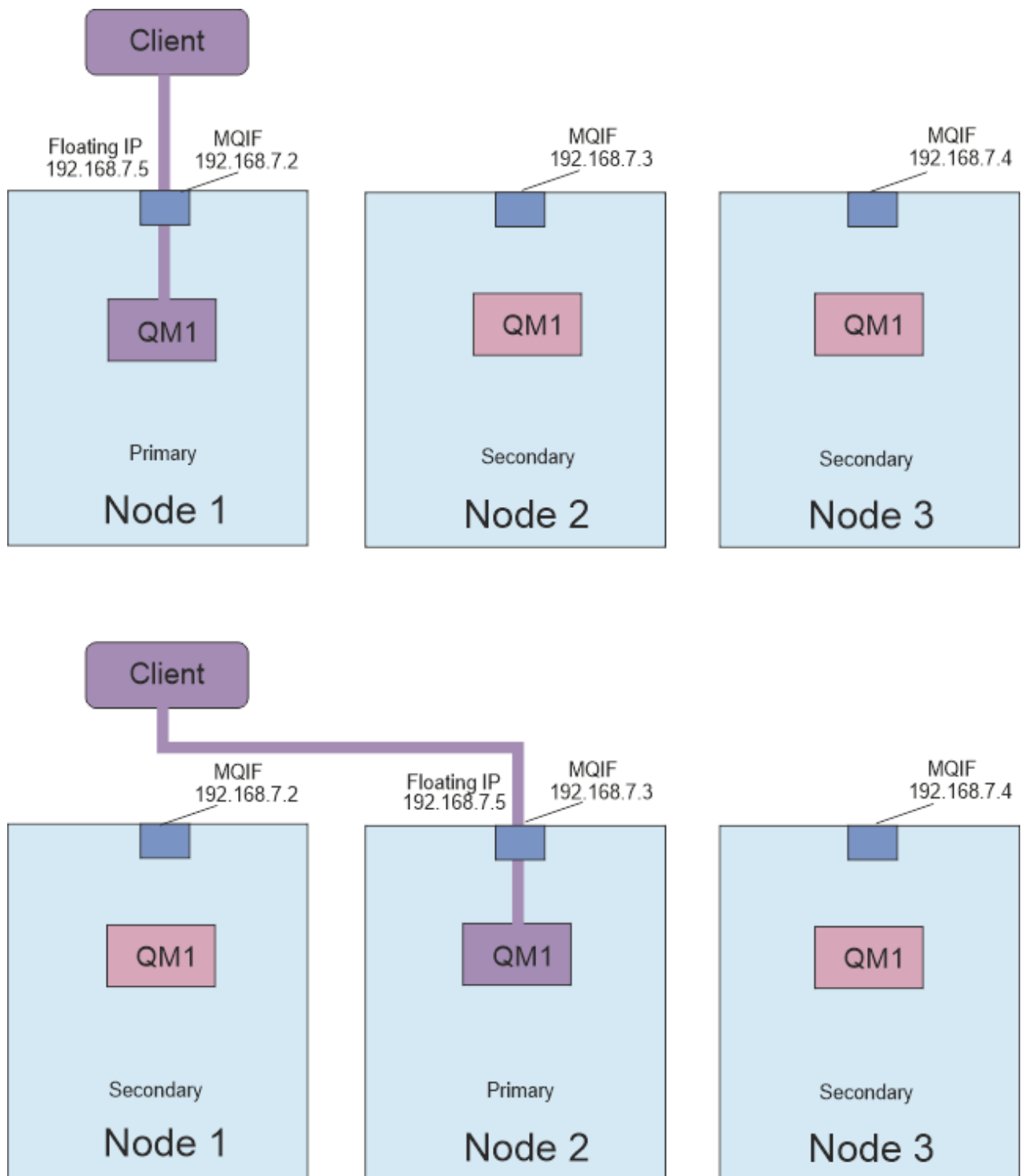


Figure 80. Floating IP address

You must be a user in both the `mqm` and `haclient` groups to run the **rdqmint** command. You can create or delete the floating IP address on the primary node for the RDQM, or either of the secondary nodes.

Note: You cannot use the same floating IP address for multiple RDQMs, the floating IP address for each RDQM must be unique.

Procedure

- To create a floating IP address for an RDQM, enter the following command:

```
rdqmint -m qmname -a -f ipv4address -l interfacename
```

where:

qmname

Is the name of the RDQM you are creating the floating IP address for.

ipv4address

The floating IP address in ipv4 format.

The floating IP address must be a valid IPv4 address that is not already defined on any HA node, and it must belong to the same subnet as the static IP addresses defined for the local interface.

interfacename

The name of the physical interface on the primary node to bind to.

For example:

```
rdqmint -m QM1 -a -f 192.168.7.5 -l MQIF
```

- To delete an existing floating IP address, enter the following command:

```
rdqmint -m qmname -d
```

Related reference

[rdqmint \(add or delete floating IP address for RDQM\)](#)

Starting, stopping, and displaying the state of an HA RDQM

You use variants of standard IBM MQ control commands to start, stop, and view the current state of a replicated data queue manager (RDQM).

About this task

You must run the commands that start, stop, and view the current state of a replicated data queue manager (RDQM) as a user that belongs to both the `mqm` and `haclient` groups.

You must run the commands to start and stop a queue manager on the primary node for that queue manager.

Procedure

- To start an RDQM, enter the following command on the RDQM's primary node:

```
strmqm qmname
```

where *qmname* is the name of the RDQM that you want to start.

The RDQM is started, and Pacemaker starts managing the RDQM. You must specify the `-ns` option with `strmqm` if you want to specify any other `strmqm` options.

- To stop an RDQM, enter the following command on the RDQM's primary node:

```
endmqm qmname
```

where *qmname* is the name of the RDQM that you want to stop.

Pacemaker ceases to manage the RDQM, and then the RDQM is ended. All other **endmqm** parameters can be used when stopping an RDQM.

- To view the state of an RDQM, enter the following command:

```
dspm
```

The state information that is output depends on whether you run the command on the RDQM's primary or secondary node. If run on the primary node then one of the normal status messages returned by **dspmq** is displayed. If you run the command on a secondary node then the status `running elsewhere` is displayed. For example, if **dspmq** is run on node RDQM7, the following information might be returned:

```
QMNAME(RDQM8)      STATUS(Running elsewhere)
QMNAME(RDQM9)      STATUS(Running elsewhere)
QMNAME(RDQM7)      STATUS(Running)
```

If the primary node is not available, or if **dspmq** is run by a user who is not `root` or a member of the `haclient` group, then the `Unavailable` state is reported. For example:

```
QMNAME(RDQM8)      STATUS(Unavailable)
QMNAME(RDQM9)      STATUS(Unavailable)
QMNAME(RDQM7)      STATUS(Unavailable)
```

You can enter the command **dspmq -o ha** (or **dspmq -o HA**) to view a list of queue managers known to a node, and whether they are RDQMs or not, for example:

```
dspmq -o ha

QMNAME(RDQM8)      HA(Replicated)
QMNAME(RDQM9)      HA(Replicated)
QMNAME(RDQM7)      HA(Replicated)
QMNAME(QM7)        HA()
```

Related reference

[dspmq \(display queue managers\)](#)

[endmqm \(end queue manager\)](#)

[strmqm \(start queue manager\)](#)

Failed resource actions

Failed resource actions arise when the Pacemaker component of an RDQM high availability configuration encounters some problem with a resource on one of the nodes in an HA group.

The RDQM HA solution uses Pacemaker for monitoring and managing resources (see [“RDQM high availability”](#) on page 622). If Pacemaker encounters an error performing an operation on a resource on a node, it records this information using a failed resource action. Some failed resource actions prevent the resource from running and must be cleared before Pacemaker can restart the resource.

You can use the **rdqmstatus -m** command to see if there any failed resource actions that are stopping a queue manager from starting on one or more nodes.

You can then use the **rdqmstatus -m qmname -a** command to view the details of failed resource actions that are associated with a queue manager. Follow this action by using the **rdqmclean** command to clear these failed resource actions, and so free up any restricted resources. (You must also take action to resolve the problems that caused the failed resource action in the first place.)

The following resources are controlled by Pacemaker in an RDQM HA configuration, and can be the subjects of failed resource actions:

- Queue manager
- Floating IP
- RDQM control
- Filesystem
- DR replication (DRBD)
- HA replication (DRBD)

Each type of resource can be subject to the following types of failure:

Soft

Soft failures are transient, and Pacemaker continues to try to recover the resource until it times out or is otherwise stopped.

Hard

A hard error requires administrative intervention. Hard errors block the resource from running on a particular node.

Fatal

A fatal error requires administrative intervention. Fatal errors block the resource from running on any node.

See [“Viewing HA RDQM and HA group status” on page 650](#) for examples of status including failed resource queue actions.

You can use the **rdqmclean** command to clear all failed resource actions associated with a specified queue manager, or all failed resource actions in the RDQM HA configuration.

Note: Some failed resource actions do not result in the queue manager from being blocked on a node. For example, after an unexpected queue manager end, Pacemaker attempts to restart the queue manager on the node on which it was found to be not running. If the start is successful, the queue manager is not blocked from running on the node. The only way you would become aware of the failed resource action in this case is by running **rdqmstatus -m qmname -a**.

Related tasks

[“Viewing HA RDQM and HA group status” on page 650](#)

You can view the status of the HA group and of individual HA replicated data queue managers (RDQMs).

Related reference

[rdqmclean](#)

[rdqmstatus](#)

Viewing HA RDQM and HA group status

You can view the status of the HA group and of individual HA replicated data queue managers (RDQMs).

About this task

You use the **rdqmstatus** command to view the status of individual RDQMs and of the HA group as a whole.

The summary status for a node also displays information about the DRBD kernel module that RDQM relies upon. When you upgrade RDQM, it is important to ensure that the correct version of the DRBD kernel module is installed for the version of RHEL kernel running on the system. The status displays the version of the OS kernel, the kernel version that the DRBD module was built for, the DRBD version, and the DRBD kernel module loaded status.

You must be a user in the **mqm** and **haclient** groups to run the **rdqmstatus** command. You can run the command on any of the three nodes.

Procedure

- To view the summary status of a node and the RDQMs that are part of the HA configuration:

```
rdqmstatus
```

The identity of the node that you ran the command on, the kernel and DRBD details for that node, and the status of the RDQMs in the HA configuration is displayed, for example:

```
Node:                               mqhavm07.exampleco.com
OS kernel version:                   5.14.0-362.18.1
DRBD OS kernel version:              5.14.0-362.18.1
DRBD version:                        9.2.7
DRBD kernel module status:           Loaded
```

```

Queue manager name:      RDQM8
Queue manager status:    Running elsewhere
HA current location:     mqhvm08.exampleco.com
HA preferred location:   mqhvm08.exampleco.com
HA blocked location:     None

Queue manager name:      RDQM9
Queue manager status:    Running elsewhere
HA current location:     mqhvm09.exampleco.com
HA preferred location:   mqhvm09.exampleco.com
HA blocked location:     None

Queue manager name:      RDQM7
Queue manager status:    Running
HA current location:     This node
HA preferred location:   This node
HA blocked location:     None

```

The DRBD kernel module status is one of the following values:

Loaded

Indicates that the DRBD module has been loaded.

Partially loaded

Can occur when the DRBD module has been loaded, but does not function correctly due to a mismatch.

Not loaded

The DRBD module is not loaded. This can be displayed on a newly installed configuration, when no RDQM queue managers have yet been created.

Not installed

Indicates that either the DRBD module is not installed, or that IBM MQ was unable to determine the OS kernel version of the DRBD module.

Previously installed version still loaded

This status can arise if a new DRBD module is installed while the existing DRBD module is running (that is, an RDQM queue manager is running). The newly installed module is reported in the status, but is not the module that is actually running.

- To view the status of the three nodes in the HA group, enter the following command:

```
rdqmstatus -n
```

The online or offline status of each node is reported. For example:

```

Node mqha04(mqhvm04.example.com) is online
Node mqha05(mqhvm05.example.com) is offline
Node mqha06(mqhvm06.example.com) is online

```

 The encryption status for group is also displayed. For example:

```
HA Group Encryption: Configured
```

or

```
HA Group Encryption: Not Configured
```

or

```
HA Group Encryption: Unknown
```

The Unknown status is returned if the **rdqmstatus -n** command is run by a non-root user that has not had **sudo** correctly configured.

- To view the status of a particular queue manager on all the nodes in the HA group, enter the following command:

```
rdqmstatus -m qmname
```

where *qmname* is the name of the RDQM you want to view the status for. The status of the RDQM on the current node is displayed, followed by a summary of the status of the other two nodes from the perspective of the current node.

- To view the status of a particular queue manager on all the nodes in the HA group, including details of any failed resource actions, enter the following command:

```
rdqmstatus -m qmname -a
```

where *qmname* is the name of the RDQM you want to view the status for. The status of the RDQM on the current node is displayed, followed by a summary of the status of the other two nodes from the perspective of the current node. This is followed by details about any failed resource actions associated with the RDQM.

- Summary

The following table summarizes the information about the current node that can be returned by the `rdqmstatus -m qmname` command for an RDQM.

Table 31. Current node status		
Status attribute	Possible values	When displayed
Node name	<i>nodename</i>	Always displayed
Queue manager status	Running Running elsewhere Ended Unavailable	Always displayed
CPU	<i>n.nn%</i>	Only shown when current node has primary role (that is, the RDQM is running on this node)
Memory	<i>nnn</i> MB used, <i>y.y</i> GB allocated	Only shown when current node has primary role (that is, the RDQM is running on this node)
Queue manager file system	<i>nnn</i> MB used, <i>y.y</i> GB allocated [<i>z%</i>]	Only shown when current node has primary role (that is, the RDQM is running on this node)
HA role	Primary Secondary Unknown	Always displayed
HA status	All nodes in standby This node in standby Remote nodes in standby Mixed <i>status of remote nodes</i>	All nodes in standby Current node in standby Both remote nodes in standby Different status for each remote node (see next table for individual status) Same status for both remote nodes (see next table for all values)
HA preferred location	None This node Unknown <i>nodename</i>	Always displayed

Table 31. Current node status (continued)		
Status attribute	Possible values	When displayed
HA blocked location	None - The queue manager is not blocked from running on any nodes This node - The queue manager is blocked from running on the current node due to one or more failed resource actions <i>nodename</i> - The queue manager is blocked from running on <i>nodename</i> due to one or more failed resource actions <i>nodename1, nodename2</i> - The queue manager is blocked from running on <i>nodename1</i> and <i>nodename2</i> due to one or more failed resource actions All nodes - The queue manager is blocked from running on all nodes due to one or more failed resource actions	Always displayed
HA floating IP interface	<i>Interface_name</i>	Always displayed
HA floating IP address	<i>IPV4_address</i>	Always displayed

The following table summarizes the information that is returned by the `rdqmstatus -m qmname` command for the other nodes in the HA group.

Table 32. Other node status		
Status attribute	Possible values	When displayed
Node name	<i>nodename</i>	Always displayed
HA status	Normal Synchronization in progress Remote unavailable Inconsistent Paused Remote node in standby Unknown	Nodes are in sync with each other Synchronizing with remote node Unable to communicate with remote node Out of sync with remote node, and not synchronizing Replication paused Remote node in standby
 HA TLS encryption	Yes No Not configured	Always displayed. Indicates whether the HA replication connection is configured to be secured with TLS, and if it is, whether the configuration is active between the local node and the remote node.
HA synchronization in progress	<i>n.n%</i>	Displayed when synchronization in progress, and command run as root

Table 32. Other node status (continued)

Status attribute	Possible values	When displayed
HA estimated synchronization time	yyyy-mm-dd hh:mm:ss.nnn	Displayed when synchronization in progress
HA out of sync data	nKB	Displayed when remote node unavailable or inconsistent
HA last in sync	yyyy-mm-dd hh:mm:ss.nnn	Displayed when HA data is out of sync (after initial synchronization). Gives the time and date when the data was last in sync.

Example

Example of normal status on primary node:

```

Node:                               mqhavam07.exampleco.com
Queue manager status:               Running
CPU:                               0.00
Memory:                             123MB
Queue manager file system:          606MB used, 1.0GB allocated [60%]
HA role:                             Primary
HA status:                           Normal
HA control:                           Enabled
HA current location:                 This node
HA preferred location:                This node
HA preferred location:                This node
HA blocked location:                 None
HA floating IP interface:             eth4
HA floating IP address:              192.0.2.4

Node:                               mqhavam08.exampleco.com
HA status:                           Normal
HA TLS encryption:                   Not configured

Node:                               mqhavam09.exampleco.com
HA status:                           Normal
HA TLS encryption:                   Not configured

```

Example of normal status on a secondary node:

```

Node:                               mqhavam08.exampleco.com
Queue manager status:               Running elsewhere
HA role:                             Secondary
HA status:                           Normal
HA control:                           Enabled
HA current location:                 mqhavam07.exampleco.com
HA preferred location:                mqhavam07.exampleco.com
HA blocked location:                 None
HA floating IP interface:             eth4
HA floating IP address:              192.0.2.4

Node:                               mqhavam07.exampleco.com
HA status:                           Normal
HA TLS encryption:                   Not configured

Node:                               mqhavam09.exampleco.com
HA status:                           Normal
HA TLS encryption:                   Not configured

```

Example of status on primary node when synchronization is in progress:

```

Node:                               mqhavam07.exampleco.com
Queue manager status:               Running
CPU:                               0.53
Memory:                             124MB
Queue manager file system:          51MB used, 1.0GB allocated [5%]
HA role:                             Primary
HA status:                           Synchronization in progress

```

```

HA control: Enabled
HA current location: This node
HA preferred location: This node
HA blocked location: None
HA floating IP interface: eth4
HA floating IP address: 192.0.2.4

Node: mqhvm08.exampleco.com
HA status: Synchronization in progress
HA TLS encryption: Not configured
HA synchronization progress: 11.0%
HA estimated time to completion: 2017-09-06 14:55:05

Node: mqhvm09.exampleco.com
HA status: Synchronization in progress
HA TLS encryption: Not configured
HA synchronization progress: 11.0%
HA estimated time to completion: 2017-09-06 14:55:06

```

Example of status on primary node when synchronization has been lost:

```

Node: mqhvm07.exampleco.com
Queue manager status: Running
CPU: 0.53
Memory: 124MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
HA role: Primary
HA status: Mixed
HA control: Enabled
HA current location: This node
HA preferred location: This node
HA blocked location: None
HA floating IP interface: eth4
HA floating IP address: 192.0.2.4

Node: mqhvm08.exampleco.com
HA status: Normal
HA TLS encryption: Not configured

Node: mqhvm09.exampleco.com
HA status: Inconsistent
HA TLS encryption: Not configured
HA out of sync data: 15932KB
HA last in sync: 2017-09-06 14:55:06

```

Example of a primary node showing multiple states:

```

Node: mqhvm07.exampleco.com
Queue manager status: Running
CPU: 0.02
Memory: 124MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
HA role: Primary
HA status: Mixed
HA control: Enabled
HA current location: This node
HA preferred location: This node
HA blocked location: None
HA floating IP interface: eth4
HA floating IP address: 192.0.2.4

Node: mqhvm08.exampleco.com
HA status: Normal
HA TLS encryption: Not configured

Node: mqhvm09.exampleco.com
HA status: Inconsistent
HA TLS encryption: Not configured

```

Example of a primary node showing failed resource actions:

```

Node: mqhvm07.exampleco.com
Queue manager status: Running
CPU: 0.00%
Memory: 123MB
Queue manager file system: 606MB used, 1.0GB allocated [60%]
HA role: Primary
HA status: Normal

```

```

HA control: Enabled
HA current location: This node
HA preferred location: mqhavam08.exampleco.com
HA blocked location: mqhavam08.exampleco.com
HA floating IP interface: eth4
HA floating IP address: 192.0.2.4

Node: mqhavam08.exampleco.com
HA status: Normal
HA TLS encryption: Not configured

Node: mqhavam09.exampleco.com
HA status: Normal
HA TLS encryption: Not configured

Failed resource action: Start
Resource type: Filesystem
Failure node: mqhavam08.exampleco.com
Failure time: 2017-09-06 12:00:00
Failure reason: Couldn't find directory [/var/mqm/vols/qmname] to use
as a mount point
Blocked location: mqhavam08.exampleco.com

```

This status shows that Pacemaker was unable to start the filesystem on node mqhavam08.exampleco.com at 12:00:00. This failed resource action means that the queue manager is blocked from running on mqhavam08.exampleco.com. After the underlying issue that caused the failed resource action is resolved, run the **rdqmclean** command to clear the failed action so that Pacemaker can retry the action (if required).

Example of a summary status showing a mismatch between the OS kernel version (RHEL 9.3) and the DRBD kernel module (targeted for RHEL 9.2). Even though the status reports that DRBD kernel module is loaded and queue manager is running, you should update DRBD kernel module with the version targeted for the running OS kernel in this situation.

```

Node: mqhavam07.exampleco.com
OS kernel version: 5.14.0-362.18.1
DRBD OS kernel version: 5.14.0-284.11.1
DRBD version: 9.2.7+ptf.14
DRBD kernel module status: Loaded

Queue manager name: RDQM7
Queue manager status: Running
HA current location: This node
HA preferred location: This node
HA blocked location: None

```

Example of a summary status showing a mismatch between the OS kernel version (RHEL 8.10) and the DRBD kernel module (targeted for RHEL 8.8). In this example the version mismatch is more severe and DRBD kernel module fails to load successfully. As the result the queue manager fails to start on its preferred node and its HA status is Unknown. To resolve this failure the DRBD kernel module must be updated with the version target for the running OS kernel.

```

Node: mqhavam57.exampleco.com
OS kernel version: 4.18.0-553
DRBD OS kernel version: 4.18.0-477
DRBD version: 9.2.7+ptf.14
DRBD kernel module status: Partially loaded

Queue manager name: QM2
Queue manager status: Running elsewhere
HA status: Unknown
HA current location: mqhavam58.exampleco.com
HA preferred location: This node
HA blocked location: All nodes

```

Related reference

 [rdqmstatus](#)

Changing IP addresses in high availability configurations

If you change the IP addresses of any of the interfaces in a high availability configuration, high availability operation is no longer available and the queue manager will not run on the node where the addresses were changed.

You specify up to three IP addresses for HA operation in the `rdqm.ini` file. If you have already changed the Pacemaker monitor addresses, you must temporarily restore them to their original values before following the procedure. Otherwise it is not possible to delete the HA RDQM queue manager.

1. Remove the HA configuration on each node. You remove HA by backing up the queue managers and then deleting them, see [“Backing up and restoring IBM MQ queue manager data” on page 759](#) and [“Deleting an HA RDQM” on page 639](#), and then removing the HA group itself, see [“Deleting the Pacemaker cluster \(HA group\)” on page 633](#).
2. Re-create the HA configuration with the new IP addresses, see [“Defining the Pacemaker cluster \(HA group\)” on page 630](#).
3. Recreate the HA queue managers and restore the back up, see [“Creating an HA RDQM” on page 637](#) and [“Backing up and restoring IBM MQ queue manager data” on page 759](#).

Replacing a failed node in a high availability configuration

If one of the nodes in your HA group fails, you can replace it.

About this task

The steps to take to replace a node depend on the scenario:

- If you are replacing the failed node with a node with an identical configuration, you can replace the node without disrupting the HA group.
- If the new node has a different configuration, then you must delete and then rebuild the HA group. You can first back up the queue managers from the node they are running on, then restore them after you have rebuilt the HA group.

Procedure

- If the replacement node is configured to look like the failed node (same hostname, same IP addresses, and so on), then complete the following steps on the new node:
 - a) Create an `rdqm.ini` file that matches the files on the other nodes, and then run the `rdqmadm -c` command (see [“Defining the Pacemaker cluster \(HA group\)” on page 630](#)).
 - b) Run the `crtmqm -sxs qmanager` command to recreate each replicated data queue manager (see [“Creating an HA RDQM” on page 637](#)).
- If the replacement node has a different configuration to the failed node:
 - a) If required, back up your queue managers (see [“Backing up and restoring IBM MQ queue manager data” on page 759](#)).
 - b) Delete the replicated data queue managers from the other nodes in the HA group by using the `dlrmqm` command (see [“Deleting an HA RDQM” on page 639](#)).
 - c) Unconfigure the Pacemaker cluster by using the `rdqmadm -u` command (see [“Deleting the Pacemaker cluster \(HA group\)” on page 633](#)).
 - d) Reconfigure the Pacemaker cluster, including the information for the new node, by using the `rdqmadm -c` command (see [“Defining the Pacemaker cluster \(HA group\)” on page 630](#)).
 - e) If necessary (that is, if you do not have SSH access to the other nodes) run the `crtmqm -sxs qmanager` command to recreate each replicated data queue manager on the other nodes (see [“Creating an HA RDQM” on page 637](#)).
 - f) Run the `crtmqm -sx qmanager` command to create the queue managers on the replacement node.

- g) If required, restore data and configuration to your queue managers (see [“Backing up and restoring IBM MQ queue manager data”](#) on page 759).

Linux MQ Adv. RDQM disaster recovery

RDQM (replicated data queue manager) is available on a subset of Linux platforms and can provide a disaster recovery solution.

See [Software Product Compatibility Reports](#) for full details.

You can create a primary instance of a disaster recovery queue manager running on one server, and a secondary instance of the queue manager on another server that acts as the recovery node. Data is replicated between the queue manager instances. If you lose your primary queue manager, you can manually make the secondary instance into the primary instance and start the queue manager, then resume work from the same place. You cannot start a queue manager while it is in the secondary role. The replication of the data between the two nodes is handled by DRBD.

You can choose between synchronous and asynchronous replication of data between primary and secondary queue managers. If you select the asynchronous option, operations such as IBM MQ PUT or GET complete and return to the application before the event is replicated to the secondary queue manager. Asynchronous replication means that, following a recovery situation, some messaging data might be lost. But the secondary queue manager will be in a consistent state, and able to start running immediately, even if it is started at a slightly earlier part of the message stream.

You cannot add disaster recovery to an existing queue manager, although you can migrate an existing queue manager to become a RDQM queue manager (see [“Migrating a queue manager to become a DR RDQM queue manager”](#) on page 668).

You can have several pairs of RDQM queue managers running on a number of different servers. For example, you could have primary disaster recovery queue managers running on different nodes, while all their secondary disaster recovery queue managers run on the same node. Some example configurations are illustrated in the following diagrams.

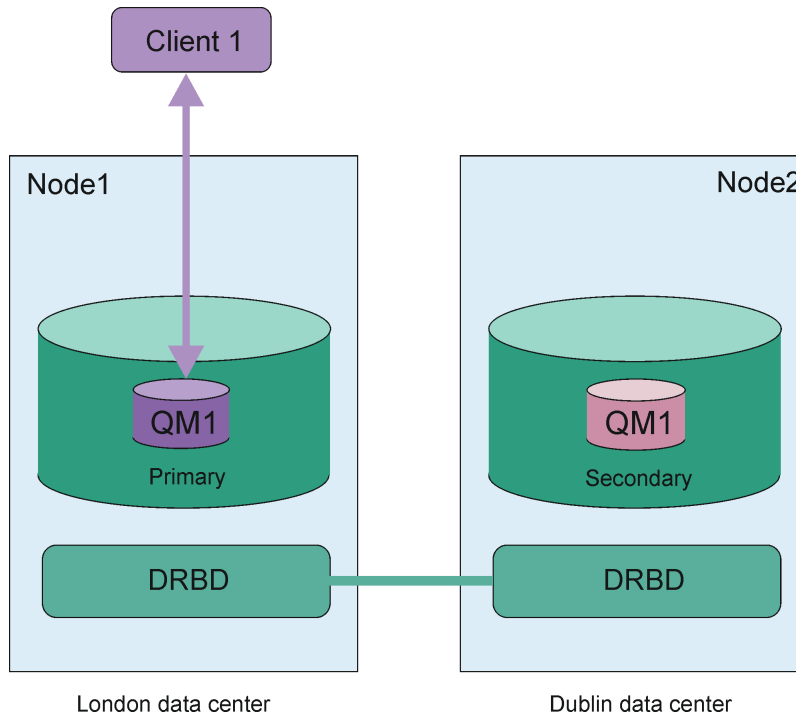


Figure 81. Single RDQM pair

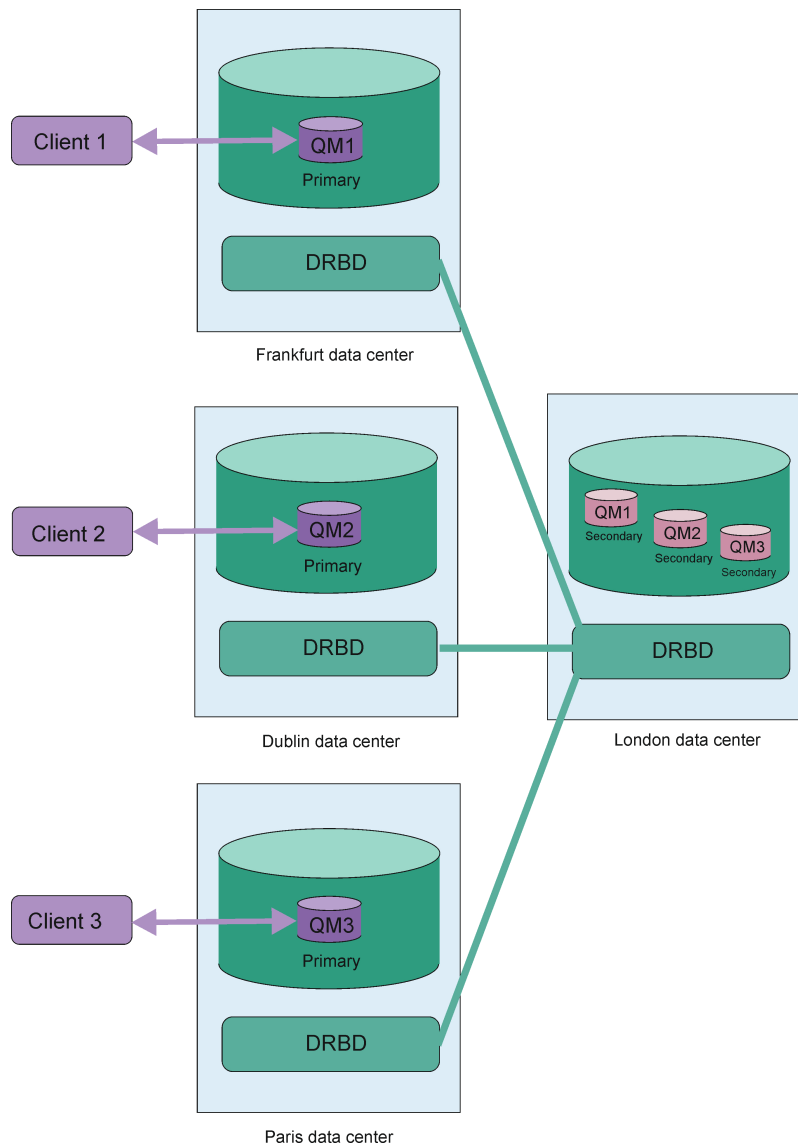


Figure 82. Secondary queue managers in same node

Replication, synchronization, and snapshots

While the two nodes in a disaster recovery configuration are connected, any updates to the persistent data for a disaster recovery queue manager are transferred from the primary instance of the queue manager to the secondary instance. This is known as **replication**.

If the network connection between the two nodes is lost, the changes to the persistent data for the primary instance of a queue manager are tracked. When the network connection is restored, a different process is used to get the secondary instance up to speed as quickly as possible. This is known as **synchronization**.

While synchronization is in progress, the data on the secondary instance is in an inconsistent state. A **snapshot** of the state of the secondary queue manager data is taken. If a failure of the main node or the network connection occurs during synchronization, the secondary instance reverts to this snapshot and the queue manager can be started. Any of the updates that happened since the original network failure are lost, however.

Partitioned data (split brain)

DR RDQM configurations require user action after loss of the primary instance of a queue manager to promote and run the secondary instance on the recovery node. It is the responsibility of whoever (or whatever) promotes the secondary instance to ensure that the former primary queue manager is stopped. If the original primary keeps running, it might process messages and, when normal operation is restored, the two instances of the queue manager have different views of the data. This is known as a partitioned or split-brain state.

Consider the following situations:

- The node on which the primary queue manager is running fails completely. You promote the secondary instance to become the primary; you cannot take action to stop the original primary because it is not running. When the original node is repaired or replaced, the queue manager on that node will be initially made the secondary and be synchronized with the primary queue manager on the recovery node. The roles of the two queue managers are then reversed, and normal operation recommences. The only potential data loss in this situation is any data that the primary had not completed replicating to the secondary before the node failed.
- There is a network failure affecting the replication link between the nodes running the primary and secondary instances of the queue manager. In this situation you must ensure that you stop the original primary before you promote the secondary. If the original primary still has other network connectivity you effectively have two primary instances running at the same time, and partitioned data can accrue. (If the replication link is working, you cannot promote a secondary queue manager if the primary instance is still running, the command fails.)
- There is a complete network failure on the node running the primary instance of the queue manager. Again you must ensure that you stop the primary instance before you promote the secondary. If the previous primary is still running when the network is restored, there will be two primary instances, and again partitioned data will accrue.

When you do a managed failover, you should not see a DR status of `partitioned` for the queue manager instances. A managed failover ends the queue manager on the primary node, then starts the queue manager on the recovery node after data has been fully replicated. A partitioned state is not expected because the queue manager is ended and data is synchronized between the nodes before it is started on the recovery node. If the queue manager is started on the recovery node while there is a loss of connectivity between the nodes then data divergence is likely if the queue manager was active on the main node when connectivity was lost. In this scenario, a partitioned state is expected to be reported once connectivity is restored because the queue manager data was not synchronized. If a partitioned state occurs you might have to examine the two data sets and make an informed decision about what set to keep. See [“Resolving a partitioned \(split brain\) problem in DR RDQM” on page 684](#).

Requirements for RDQM DR solution

You must meet a number of requirements before you configure an RDQM disaster recovery (DR) queue manager pair.

System requirements

Before you configure RDQM DR, you must complete some configuration on each of the servers that are to host RDQM DR queue managers.

- Each node requires a volume group named `d1rbdpool`. The storage for each disaster recovery replicated data queue manager (DR RDQM) is allocated as two separate logical volumes per queue manager from this volume group. (Each queue manager requires two logical volumes to support the reverting to snapshot operation, so each DR RDQM is allocated just over twice the storage that you specify when you create it.) For the best performance, this volume group should be made up of one or more physical volumes that correspond to internal disk drives (preferably SSDs).
- After you have created the `d1rbdpool` volume group, do nothing else with it. IBM MQ manages the logical volumes created in `d1rbdpool`, and how and where they are mounted.

- Each node requires an interface that is used for the data replication. This should have sufficient bandwidth to support the replication requirements given the expected workload of all of the replicated data queue managers.

For maximum fault tolerance, this interface should be an independent Network Interface Cards (NICs).

- DRBD requires that each node used for RDQM has a valid internet host name (the value that is returned by `uname -n`), as defined by RFC 952 amended by RFC 1123.
- If there is a firewall between the nodes used for DR RDQM, then the firewall must allow traffic between the nodes on the ports that are used for replication. A sample script is provided, `/opt/mqm/samp/rdqm/firewalld/configure.sh`, that opens up the necessary ports if you are running the standard firewall in RHEL. You must run the script as `root`. If you are using some other firewall, examine the service definitions `/usr/lib/firewalld/services/rdqm*` to see which ports need to be opened. The script adds the following permanent firewallD service rules for DRBD and IBM MQ (you can edit the script to omit the Pacemaker ports if you are not using HA):
 - `MQ_INSTALLATION_PATH/samp/rdqm/firewalld/services/rdqm-drbd.xml` allows TCP ports 7000-7100.
 - `MQ_INSTALLATION_PATH/samp/rdqm/firewalld/services/rdqm-mq.xml` allows TCP port 1414 (you must edit the script if you require a different port)
- If the system uses SELinux in a mode other than permissive, you must run the following command:

```
semanage permissive -a drbd_t
```

Network requirements

It is recommended that you locate the nodes used for disaster recovery in different data centers.

You should be aware of the following limitations:

- Performance degrades rapidly with increasing latency between data centers. IBM will support a latency of up to 5 ms for synchronous replication and 100 ms for asynchronous replication.
- The data sent across the replication link is not subject to any additional encryption beyond that which might be in place from using IBM MQ AMS.
- Configuring an RDQM queue manager for disaster recovery incurs an overhead due to the requirement to replicate data between the two RDQM nodes. Synchronous replication incurs a greater overhead than asynchronous replication. When synchronous replication is used, disk I/O operations are blocked until the data is written to both nodes. When asynchronous replication is used, data must only be written to the primary node before processing can continue.

Replication and network capacity

You can choose between synchronous and asynchronous replication of data between primary and secondary queue managers (DR replication is always asynchronous for DR/HA queue managers). When asynchronous replication is used, any data written on the DR primary queue manager instance must be written to the local disk of the DR primary instance and copied into the sendbuffer of the TCP connection to the DR secondary instance before the write completes on the DR primary instance. Any congestion of the DR network can limit when a write completes on the DR primary instance, and therefore the performance of a queue manager. Such congestion could cause the sendbuffer to fill to the point where new data cannot be copied into it, and the queue manager will pause persistent messaging operations until the sendbuffer has drained sufficiently to allow the waiting data to be copied into it.

Once the sendbuffer can accept the data, the asynchronous replication proceeds as usual (so it does not become true synchronous replication). It is important to ensure that the available DR network capacity is sufficient for the messaging workload.

User requirements for working with queue managers

To create, delete, or configure replicated data queue managers (RDQMs) you must either be the root user, or have a user ID belonging to the mqm group that is granted sudo authority for the following commands:

- **crtmqm**
- **dltmqm**
- **rdqmdx**

A user who belongs to the mqm group can view the state and status of a DR RDQM by using the following commands:

- **dspmq**
- **rdqmstatus**

The mqm user must have the same UID on both servers, and the mqm group must have the same GID on both servers.

Secure replication for DR RDQM

You can specify that the replication link in your DR RDQM configuration is secured with TLS.

When you create a disaster recovery (DR) queue manager in an RDQM configuration, you can specify that the link used to replicate data between the primary and secondary queue manager instances is secured with TLS.

To configure secure replication, complete the following steps:

1. Configure the required certificates.
2. Configure the **tlshd** service.
3. Specify the **-re** option when creating a new queue manager using **crtmqm**.

Complete the first two steps once per node. The certificates and **tlshd** configuration are shared by all RDQM queue managers on the node that are configured to use secure replication links.

The TLS protocols and cipher suites that can be used by the RDQM secure replication links are controlled by the active system-wide cryptographic policy. Use the Linux **update-crypto-policies** command to change the current active policy.

Configuring certificates

You must configure the certificates to be used by the replication link. The certificates that you create for each node can be used by any replication links. If you have more than one RDQM queue manager on a node that uses secure replication links, the same certificate is used.

Note: If any queue manager that runs on a node might ever be part of a DR/HA configuration, then a Subject Alternative Name (SAN) must be specified when the certificates are created. For more information, see [“Secure replication for DR/HA RDQM” on page 688](#).

To use a secure replication link in a RDQM DR configuration you need two certificates, one certificate per node in the RDQM DR group.

To configure your certificates complete the following steps for each node in your RDQM configuration:

1. Use the following command to generate a private key:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -out key_file
```

where *key_file* is the name of the file that is created to store the private key.

2. Use the following command to create a certificate signing request:

```
openssl req -key key_file -subj "/CN=hostname" -new -out cert_req_file
```

where *key_file* is the file that contains the private key that you created in step “1” on page 662, *hostname* is the hostname of the node, and *cert_req_file* is the file that is created to store the certificate signing request.

Set the certificate Common Name (CN) to the hostname of the RDQM node.

You can also specify a CN that differs from the RDQM node hostname. If you do so, ensure that the certificate is created with a Subject Alternative Name (SAN) that contains the RDQM node hostname.

3. Send the certificate signing request to your certificate authority (CA) to be signed. When you receive the signed certificate from the CA, copy the private key, signed certificate, and CA certificate files to the directory where you want to store these files. For example, the `/etc/db2d.d/` directory.
4. Set the file permissions and ownership of the private key and certificate files to ensure that only the root user can view the private keys.
 - Change the ownership of the private key and certificate file to the root user.
 - Set the permissions on the private key file so that only the root user has read and write access.

The following example script creates an internal certificate authority and two certificates for use in a RDQM DR configuration.

Important: This example is not suitable for a production environment, and is intended only as an example to set up a test environment quickly. Certificate management is a complex subject for advanced users. For production, you must consider things like rotation, revocation, key length, and much more.

Run the following script on one of the nodes:

```
#!/bin/bash

# The CA name
ca_name="RDQM Test CA"

# The CA filename
ca_filename="rdqm_test_ca"

# Host certificates to generate
test_hosts=("alice" "bob")

mkdir certificates && cd ./certificates

# Create private key for use with private CA using OpenSSL
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 \
-out $ca_filename.key

# Create a private CA certificate using OpenSSL
openssl req -key $ca_filename.key -new -x509 -days 3650 -subj "/CN=$ca_name" \
-out $ca_filename.crt

# Create certificates for each node
for host in ${test_hosts[@]}; do

    # Create a private key
    openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 \
    -out $host.key

    # Generate a CSR
    openssl req -key $host.key -subj "/CN=$host" -new -out $host.csr

    # Create a valid certificate signed by the private CA
    openssl x509 -req -in $host.csr -CA $ca_filename.crt \
    -CAkey $ca_filename.key -CAcreateserial -days 365 -out $host.crt

    # Remove sequential serial number
    rm -f $ca_filename.srl

    # Remove CSR
    rm -f $host.csr

done
```

The example generates the following files in the certificates directory:

- `rdqm_test_ca.crt`

- `rdqm_test_ca.key`
- `alice.crt`
- `alice.csr`
- `alice.key`
- `bob.crt`
- `bob.csr`
- `bob.key`

Deploy the certificates by completing the following steps:

1. Copy the CA certificate (`rdqm_test_ca.crt`) to each node's `/etc/drbd.d/` directory. (These are example paths, you can choose another location if required.)
2. Copy each node's certificate (`.crt` file) and private key (`.key` file) to `/etc/drbd.d/`. (These are example paths, you can choose another location if required.)
3. Change file permissions and ownership for the private keys and certificate on each node. This ensures that only the root user can view the contents of the private keys:

```
chmod 600 /etc/drbd.d/*.key
chown root:root /etc/drbd.d/*.key
chown root:root /etc/drbd.d/*.crt
```

4. Store the certificate authority's private key in a secure location.

Configuring the tlshd service

This `tlshd` service needs to be configured and then enabled and started, before any DRBD TLS handshakes can be completed. The `tlshd` service is provided in a package that is installed with the `drbd-utils-9` packages.

The configuration file for this service is `/etc/tlshd.conf` and the initial contents are:

```
[debug]
loglevel=0
tls=0
nl=0

[authenticate]
#keyrings= <keyring>;<keyring>;<keyring>

[authenticate.client]
#x509.truststore= <pathname>
#x509.certificate= <pathname>
#x509.private_key= <pathname>

[authenticate.server]
#x509.truststore= <pathname>
#x509.certificate= <pathname>
#x509.private_key= <pathname>
```

Before enabling and starting the `tlshd` service, you must specify values for the following properties in both the `authenticate.client` and `authenticate.server` sections (DRBD can establish its connections in either direction, unless there is a firewall blocking one direction):

x509.truststore

The path to the file that contains the certificate of the CA that signed the certificates used by the RDQM nodes. If a complex trust chain is required, it must be captured in a single file.

x509.certificate

The path to the file that contains the CA-signed certificate for the RDQM node.

x509.private_key

The path to the file that contains the private key for the RDQM node. The file must be owned by root and read/write permissions must be granted to only the root user.

The paths that are specified in these properties must all identify local files.

You can also optionally specify a certificate revocation list (CRL). You must set this in both the `authenticate.client` and `authenticate.server` sections:

x509.crl

The path to a file containing PEM-encoded certificate revocation lists (CRL) that are to be used to verify the revocation status of certificates during each handshake. If this option is not specified, CRL checking is skipped.

If you created the certificates using the example script in [“Configuring certificates”](#) on page 662, then you would configure the file as follows for server alice:

```
[authenticate.client]
x509.truststore=/etc/drbd.d/rdqm_test_ca.crt
x509.certificate=/etc/drbd.d/alice.crt
x509.private_key=/etc/drbd.d/alice.key

[authenticate.server]
x509.truststore=/etc/drbd.d/rdqm_test_ca.crt
x509.certificate=/etc/drbd.d/alice.crt
x509.private_key=/etc/drbd.d/alice.key
```

Before you enable the service for the first time, enter the following command:

```
systemctl daemon-reload
```

To start the `tlshd` service, enter the following command:

```
systemctl enable --now tlshd
```

Creating the DR RDQM

You create DR queue manager with a secure replication link by using the **crtmqm** command with the `-re` option, for example, for the primary instance:

```
crtmqm -rr p -rl <Local_IP> -ri <Recovery_IP> -rn <Recovery_name> -rp <Port> -re <DRQMname>
```

After the command completes, it outputs the command that you need to input on the secondary node to create the secondary instance of the queue manager.

Use the **rdqmstatus** command to confirm that the replication link is secured with TLS. For more information, see [“Viewing DR RDQM status”](#) on page 675.

For more information about creating a DR RDQM, see [“Creating a disaster recovery RDQM”](#) on page 665.

Creating a disaster recovery RDQM

You use the **crtmqm** command to create a replicated data queue manager (RDQM) to act as a primary or a secondary in a disaster recovery configuration.

About this task

You can create a replicated data queue manager (RDQM) as a user in the `mqm` group if the user can use `sudo`. Otherwise you must create the RDQM as root.

You must create a primary RDQM DR queue manager on one node. Then you must create a secondary instance of the same queue manager on another node. The primary and secondary instances must have the same name and be allocated the same amount of storage.

The following points give some guidance about sizing the queue manager file system:

1. When creating an RDQM queue manager, a file system is allocated to store queue manager data and logs. It is important to size this file system appropriately so that the queue manager can record ongoing activity in its logs, and store application messages on queues. When sizing the file system, consider peak messaging requirements, future workload growth, and application outages that might cause messages to build up on queues. For guidance on calculating the size of the queue manager

recovery log, see [“How large should I make my log filesystem?”](#) on page 737. When calculating the storage requirements for application messages, the size and number of messages needs to be considered, plus their MQMD header and any message properties they have.

2. RDQM queue manager file systems cannot be dynamically resized. You must back up then restore an RDQM queue manager with a larger file system if this is required, see [“Resizing the filesystem for an HA RDQM queue manager”](#) on page 643.
3. You can limit the size of individual queues on disk by using local queue attributes, such as MAXDEPTH and MAXFSIZE. See [Modifying IBM MQ queue files](#).
4. You should monitor your ongoing disk usage and respond appropriately if disk usage increases before file system usage becomes critical. File system usage can be monitored by using either platform/operating system capabilities, or by subscribing to metrics published to IBM MQ system topics that are described at [Metrics published on the system topics](#).

V 10.0.0 You can, if required, specify that the replication link between the instances of the disaster recovery RDQM is secured with TLS. To use a secure replication link, you must configure the certificates and tlshd service before the DR RDQM is created. For more information, see [“Secure replication for DR RDQM”](#) on page 662.

Procedure

- To create a primary DR RDQM:
 - a) Enter the following command:

```
crtmqm -rr p [-rt (a | s)] -rl Local_IP -ri Recovery_IP -rn Recovery_Name -rp Port [-re]
[other_crtmqm_options] [-fs size] QMname
```

where:

-rr p

Specifies that you are creating the primary instance of the queue manager.

-rt a | s

-rt s specifies that the DR configuration uses synchronous replication, **-rt a** specifies that the DR configuration uses asynchronous replication. Asynchronous replication is the default.

-rl Local_IP

Specifies the local IP address to be used for DR replication of this queue manager.

-ri Recovery_IP

Specifies the IP address of the interface used for replication on the server hosting the secondary instance of the queue manager.

-rn Recovery_Name

Specifies the name of the system that is hosting the secondary instance of the queue manager. The name is that value that is returned if you run `uname -n` on that server. You must explicitly create a secondary queue manager on that server.

-rp Port

Specifies the port to use for DR replication.

V 10.0.0 **-re**

Specifies that the replication link between the disaster recovery RDQM instances is secured with TLS. This parameter is optional. If you specify this option for the primary instance, you must also specify it for the secondary instances.

other_crtmqm_options

You can optionally specify one or more of these general **crtmqm** options:

- -z
- -q
- -c *Text*
- -d *DefaultTransmissionQueue*

- -h *MaxHandles*
- -g *ApplicationGroup*
- -oa *user|group*
- -t *TrigInt*
- -u *DeadQ*
- -x *MaxUMsgs*
- -lp *LogPri*
- -ls *LogSec*
- -lc | -l
- -lla | -lln
- -lf *LogFileSize*
- -p *Port*

For more information about these parameters, see [crtmqm \(create queue manager\)](#).

-fs size

Optionally specifies the size of the filesystem to create for the queue manager, that is, the size of the logical volume which is created in the drbdpool volume group. Another logical volume of that size is also created, to support the reverting to snapshot operation, so the total storage for the DR RDQM is just over twice that specified here.

Size is a numeric value, which is specified in GB. You can specify a value in MB by entering the value followed by the character M. For example, to specify a filesystem size of 3 GB, enter 3. To specify a filesystem size of 1024 MB, enter 1024M. (You can also add a G suffix to explicitly state GB.)

QMname

Specifies the name of the replicated data queue manager. The name is case sensitive.

After the command completes, it outputs the command that you need to input on the secondary node to create the secondary instance of the queue manager. You can also use the **rdqmdx** command on your primary node to retrieve the **crtmqm** command that you need to run on the secondary node to create the secondary queue manager, see [“Managing primary and secondary characteristics of DR RDQMs”](#) on page 674.

- To create a secondary DR RDQM, run the **crtmqm** command that was output by the **crtmqm** command when you created the primary. You run the command on the node that is to host secondary instances of the RDQM.

What to do next

After you have created your primary and secondary instances of your queue manager, you must check the status on both nodes to check both are correct. Use the **rdqmstatus** command on both nodes. The nodes should be displaying normal status as described in [“Viewing DR RDQM status”](#) on page 675. If they are not displaying this status, delete the secondary instance and recreate it, taking care to use the correct arguments.

Related reference

[crtmqm](#)

Deleting a DR RDQM

You use the **dlrmqm** command to delete a disaster recovery replicated data queue manager (RDQM).

About this task

You must run the command to delete the RDQM on both the RDQM's primary and secondary nodes. The RDQM must be ended first. You can run the command as an mqm user if that user has the necessary sudo privileges. Otherwise, you must run the command as root.

Procedure

- To delete a DR RDQM, enter the following command:

```
dltmqm RDQM_name
```

Related reference

[dltmqm](#)

Migrating a queue manager to become a DR RDQM queue manager

You can migrate an existing queue manager to become a disaster recovery (DR) replicated data queue manager (RDQM) by backing up its persistent data, then restoring the data to a newly created RDQM queue manager that has the same name.

About this task

DR Replicated Data Queue Managers require a dedicated logical volume (filesystem) and the configuration of disk replication. These components are only configured when a new queue manager is created. An existing queue manager can be migrated to use RDQM by backing up its persistent data, then restoring the data to a newly created RDQM queue manager that has the same name. This procedure preserves the queue manager configuration, state, and persistent messages at the time the backup was created.

Note: You can only migrate a queue manager from a version of IBM MQ that is the same as or lower than the version where RDQM is installed. The operating system and architecture must also be the same. Otherwise, you must create a new queue manager on your target platform, see [Moving a queue manager to a different operation system](#).

You should satisfy the following conditions before you migrate a queue manager:

- Evaluate your disaster recovery requirements and see “[RDQM disaster recovery](#)” on page 658.
- Review the applications and queue managers that connect to the queue manager. Consider the changes required to route the connections to the RDQM node where the queue manager is running.
- Provision, or identify existing, RDQM nodes for your chosen configuration. For information about the system requirements for RDQM, see “[Requirements for RDQM DR solution](#)” on page 660.
- Install IBM MQ Advanced, which includes the RDQM feature, on each node.
- Optionally, verify the RDQM configuration using a test queue manager, which can then be deleted. Testing the configuration is recommended to identify and resolve any problems before migrating the queue manager.
- Review the security configuration for the queue manager, then replicate the required local users and groups on each RDQM node.
- Review the queue manager and channel configuration to determine if API exits, channel exits, or data conversion exits are used. Install the required exits on each RDQM node.
- Review any queue manager services that have been defined, then install and configure the required processes on each RDQM node.

Procedure

- Back up the existing queue manager:
 - Stop the existing queue manager by issuing a wait shutdown command `endmqm -w`, or an immediate shutdown command `endmqm -i`. This step is important to ensure the data in the backup is consistent.
 - Determine the location of the queue manager data directory by viewing the IBM MQ configuration file, `mqs.ini`. On Linux, this file is located in the `/var/mqm` directory. For more information about `mqs.ini`, see “[IBM MQ configuration file, mqs.ini](#)” on page 96.

Locate the QueueManager stanza for the queue manager in the file. If the stanza contains a key named DataPath then its value is the queue manager data directory. If the key does not exist, then the queue manager data directory can be determined using the values of the Prefix and Directory keys. The queue manager data directory is a concatenation of these values, of the form *prefix/qmgrs/directory*. For more information about the QueueManager stanza, see [“QueueManager stanza of the mq.ini file”](#) on page 106.

- c) Create a backup of the queue manager data directory. On Linux, you can do this by using the **tar** command. For example, to back up the data directory for a queue manager you can use the following command. Note the last parameter of the command, which is a single period (dot):

```
tar -cvzf qm-data.tar.gz -C queue_manager_data_dir .
```

- d) Determine the location of the queue manager log directory by viewing the IBM MQ queue manager configuration file `qm.ini`. This file is located in the queue manager data directory. For more information about the file, see [“Queue manager configuration files, qm.ini”](#) on page 108.

The queue manager log directory is defined as the value of the LogPath key in the Log stanza. For information about the stanza, see [“Log stanza of the qm.ini file”](#) on page 142.

- e) Create a backup of the queue manager log directory. On Linux, you can do this by using the **tar** command. For example, to back up the log directory for a queue manager you can use the following command. Note the last parameter of the command, which is a single period (dot):

```
tar -cvzf qm-log.tar.gz -C queue_manager_log_dir .
```

- f) Create a backup of any certificate repositories used by the queue manager if they are not located in the queue manager data directory. Ensure that both the key database file and the password stash file are backed up. For information about the queue manager key repository, see [The SSL/TLS key repository](#) and [Locating the key repository for a queue manager](#). For information about locating the AMS key store if the queue manager is configured to use AMS Message Channel Agent (MCA) interception, see [Message Channel Agent \(MCA\) interception](#).
- g) The existing queue manager is no longer required, so it can be deleted. However, where possible, you should only delete the existing queue manager after it has been successfully restored on the target system. Deferring deletion ensures that the queue manager can be restarted if the migration process does not complete successfully.

Note: If you defer deletion of the existing queue manager, do not restart it. It is important the queue manager remains ended because further changes to its configuration or state are lost during the migration.

2. Prepare the primary RDQM node:

- a) Create a new RDQM queue manager with the same name as the queue manager that you backed up. Ensure the filesystem allocated for the RDQM queue manager by **crtmqm** is big enough to contain the data, primary logs and secondary logs for the existing queue manager, plus some additional space for future expansion. For information about how to create an RDQM queue manager, see [“Creating a disaster recovery RDQM”](#) on page 665.
- b) Determine the primary RDQM node for the queue manager. For information about how to determine the primary node, see [rdqmstatus \(display RDQM status\)](#).
- c) On the primary RDQM node, if the RDQM queue manager is started, stop it by using the `endmqm -w` or `endmqm -i` command.
- d) Determine the location of the data and log directories for the RDQM queue manager (use the methods described in steps 1b and 1d).
- e) Delete the contents of the RDQM queue manager data and log directories, but not the directories themselves.

3. Restore the queue manager on the primary RDQM node:

- a) Copy the backups of the queue manager data and log directories to the primary RDQM node, plus any separate backups of certificate repositories used by the queue manager.

- b) Restore the backup of the queue manager data directory to the empty data directory for the new RDQM queue manager, ensuring that file ownership and permissions are preserved. If the backup was created using the example tar command in step 1c then the following command can be used by the root user to restore it:

```
tar -xvzpf qm-data.tar.gz -C queue_manager_data_dir
```

- c) Restore the backup of the queue manager log directory to the empty log directory for the new RDQM queue manager, ensuring that file ownership and permissions are preserved. If the backup was created using the example tar command in step 1e then the following command can be used by the root user to restore it:

```
tar -xvzpf qm-log.tar.gz -C queue_manager_log_dir
```

- d) Edit the restored queue manager configuration file, `qm.ini`, in the data directory for the RDQM queue manager. Update the value of the `LogPath` key in the `Log` stanza to specify the log directory for the RDQM queue manager.

Review other file paths that are defined in the configuration file and update them if necessary. For example, you might need to update the following paths:

- The path for error log files that are generated by diagnostic message services.
 - The path for exits that are required by the queue manager.
 - The path for switch load files if the queue manager is an XA transaction coordinator.
- e) If the queue manager is configured to use AMS Message Channel Agent (MCA) interception, copy the AMS key store to the new RDQM installation, then review and update the configuration. The key store must be available on each RDQM node, so if it is not located in the replicated filesystem for the queue manager it must be copied to each node instead. For more information, see [Message Channel Agent \(MCA\) interception](#).
- f) Verify that the queue manager is displayed by the **dspmq** command and its status is reported as ended. The following example shows sample output for an RDQM DR queue manager:

```
$ dspmq -o status -o dr
QMNAME(QM1) STATUS(Ended normally) DRRROLE(Primary)
```

- g) Verify that the restored queue manager data has been replicated to the secondary RDQM nodes by using the **rdqmstatus** command to display the status for the queue manager. The DR status should be reported as `Normal` on each node. The following example shows sample output for an RDQM DR queue manager:

```
$ rdqmstatus -m QM1
Queue manager status:           Ended normally
Queue manager file system:      51MB used, 1.0GB allocated [5%]
DR role:                        Primary
DR status:                      Normal
DR type:                        Synchronous
DR port:                        3000
DR local IP address:            192.168.20.1
DR remote IP address:           192.168.20.2
```

- h) Start the queue manager on the primary RDQM node.
- i) Connect to the queue manager and update the value of the `SSLKEYR` queue manager attribute to specify the new location of the queue manager certificate repository. By default, the value of this attribute is set to `queue_manager_data_directory/ssl/key`. The certificate repository must be located in the same location on each RDQM node. If the repository is not located in the replicated filesystem for the queue manager, then it must be copied to each node instead.
- j) Review the IBM MQ object definitions for the queue manager and update the value of object attributes that reference changed network settings, the IBM MQ installation directory, or the queue manager data directory, including the following objects:
- Local IP addresses used by listeners (`IPADDR` attribute).

- Local IP addresses used by channels (LOCLADDR attribute).
 - Local IP addresses defined for cluster-receiver channels (CONNAME attribute).
 - Local IP addresses defined for communication information objects (GRPADDR attribute).
 - System paths defined for process and service object definitions.
- k) Stop and restart the queue manager to ensure the changes become effective.
- l) Repeat step 3j for remote queue managers, plus equivalent settings for applications, that connect to the migrated queue manager, including:
- Channel connection names (CONNAME attribute).
 - Channel authentication rules that restrict inbound connections from the queue manager based on its IP address or hostname.
 - Client channel definition tables (CCDTs), domain name settings (DNS), network routing, or equivalent connection information.
- m) Perform a managed failover of the queue manager to each RDQM node to ensure the required configuration has been established successfully, see [“Switching over to a recovery node for a test scenario” on page 680](#).

Resizing the filesystem for a DR RDQM queue manager

To resize the filesystem for an existing disaster recovery (DR) replicated data queue manager (RDQM) you back up its persistent data, then restore the data to a newly created RDQM queue manager that has the same name but a filesystem of a different size.

About this task

DR replicated data queue managers require a dedicated logical volume (filesystem) and the configuration of disk replication. These components are only configured when a new queue manager is created. The filesystem cannot be resized after it has been created because it must have the same size on each node. To resize the filesystem for an existing replicated data queue manager (RDQM) you can back up its persistent data, then restore the data to a newly created RDQM queue manager that has the same name but a filesystem of a different size. This procedure preserves the queue manager configuration, state, and persistent messages at the time the backup was created.

Procedure

1. Back up the existing RDQM queue manager on the primary RDQM node:
 - a) Determine the primary RDQM node for the queue manager. For information about how to determine the primary node, see [rdqmstatus \(display RDQM status\)](#).
 - b) On the primary RDQM node, if the RDQM queue manager is started, stop it by using the **endmqm -w** or **endmqm -i** command.
 - c) Determine the location of the queue manager data directory by viewing the IBM MQ configuration file, `mqs.ini`. On Linux, this file is located in the `/var/mqm` directory. For more information about `mqs.ini`, see [“IBM MQ configuration file, mqs.ini” on page 96](#).

Locate the `QueueManager` stanza for the queue manager in the file. The queue manager data directory is the value of the key named `DataPath`. For more information about the `QueueManager` stanza, see [“QueueManager stanza of the mqs.ini file” on page 106](#).

- d) Create a backup of the queue manager data directory. On Linux, you can do this by using the **tar** command. For example, to back up the data directory for a queue manager you can use the following command. Note the last parameter of the command, which is a single period (.) character:

```
tar -cvzf qm-data.tar.gz -C queue_manager_data_dir .
```

- e) Determine the location of the queue manager log directory by viewing the IBM MQ queue manager configuration file `qm.ini`. This file is located in the queue manager data directory. For more information about the file, see [“Queue manager configuration files, qm.ini” on page 108](#).

The queue manager log directory is defined as the value of the `LogPath` key in the Log stanza. For information about the stanza, see [“Log stanza of the qm.ini file” on page 142](#).

- f) Create a backup of the queue manager log directory. On Linux, you can do this by using the **tar** command. For example, to back up the log directory for a queue manager you can use the following command. Note the last parameter of the command, which is a single period (.) character:

```
tar -cvzf qm-log.tar.gz -C queue_manager_log_dir .
```

- g) Delete the existing RDQM queue manager.

2. Restore the queue manager with a filesystem of the required size:

- a) Create a new RDQM queue manager with the same name as the queue manager that you backed up. Ensure the filesystem allocated for the RDQM queue manager by **crtmqm** is the size you require, and it is big enough to contain the data, primary logs, and secondary logs for the existing queue manager, plus some additional space for future expansion. For information about how to create an RDQM queue manager, see [“Creating a disaster recovery RDQM” on page 665](#).
- b) Determine the primary RDQM node for the queue manager. For information about how to determine the primary node, see [rdqmstatus \(display RDQM status\)](#).
- c) On the primary RDQM node, if the RDQM queue manager is started, stop it by using the **endmqm -w** or **endmqm -i** command.
- d) On the primary RDQM node, determine the new location of the data and log directories for the RDQM queue manager (use the methods described in steps 1c and 1e).
- e) On the primary RDQM node, delete the contents of the RDQM queue manager data and log directories, but not the directories themselves.
- f) On the primary RDQM node, restore the backup of the queue manager data directory to the empty data directory for the new RDQM queue manager, ensuring that file ownership and permissions are preserved. If the backup was created using the example **tar** command in step 1d then the following command can be used by the root user to restore it:

```
tar -xvzpf qm-data.tar.gz -C queue_manager_data_dir
```

- g) On the primary RDQM node, restore the backup of the queue manager log directory to the empty log directory for the new RDQM queue manager, ensuring that file ownership and permissions are preserved. If the backup was created using the example **tar** command in step 1f then the following command can be used by the root user to restore it:

```
tar -xvzpf qm-log.tar.gz -C queue_manager_log_dir
```

- h) On the primary RDQM node, edit the restored queue manager configuration file, `qm.ini`, in the data directory for the new RDQM queue manager. Update the value of the `LogPath` key in the Log stanza to specify the log directory for the new RDQM queue manager that you determined in step 2d. Review other file paths that are defined in the configuration file and update them if necessary. For example, you might need to update the following paths:
- The path for error log files that are generated by diagnostic message services.
 - The path for exits that are required by the queue manager.
 - The path for switch load files if the queue manager is an XA transaction coordinator.
- i) Verify that the queue manager is displayed by the **dspmq** command and its status is reported as ended. The following example shows sample output for an RDQM DR queue manager:

```
$ dspmq -o status -o dr
QMNAME(QM1) STATUS(Ended normally) DR(Primary)
```

- j) Verify that the restored queue manager data has been replicated to the secondary RDQM node by using the **rdqmstatus** command to display the status for the queue manager. The DR status should be reported as Normal on each node. The following example shows sample output for an RDQM DR queue manager on the primary node:

```
$ rdqmstatus -m QM1
Queue manager status:      Running
CPU:                      0.00
Memory:                   123MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
DR role:                   Primary
DR status:                 Normal
DR type:                   Synchronous
DR port:                   3000
DR local IP address:       192.168.20.1
DR remote IP address:      192.168.20.2
```

The following example shows sample output for an RDQM DR queue manager on the recovery node:

```
Queue manager status:      Ended immediately
DR role:                   Secondary
DR status:                 Normal
DR port:                   3000
DR local IP address:       192.168.20.2
DR remote IP address:      192.168.20.1
```

- k) Start the queue manager on the primary RDQM node.
- l) Perform a switchover of the queue manager to the recovery node to ensure the required configuration has been established successfully, see [“Switching over to a recovery node for a test scenario” on page 680](#).

Storing persistent application status

You can store persistent status information relating to applications along with other queue manager data.

Each IBM MQ queue manager has a dedicated filesystem for its persistent state, which includes both its queue data and recovery log. In an RDQM configuration the filesystem is backed by a logical volume that is replicated between the Linux systems (nodes). The filesystem includes a `userdata` directory that you can use to store persistent status information for your applications. So, when a replicated data queue manager moves to run on another node in your RDQM configuration, you have application context available as well as queue manager context. See [Directory content on Unix and Linux Systems](#).

If you choose to store application state in the `userdata` directory, you must be aware that data written to this location might consume the available disk space allocated to the queue manager. You must ensure that sufficient disk space remains available for the queue manager to write queue data, logs, and other persistent state information.

The `userdata` directory has `mqm` user and group ownership, and it is world readable so users can access it without needing to be in the IBM MQ administrator group (that is, `mqm`). You cannot modify the permissions of the `userdata` directory, but you can create content in it with whatever ownership and permissions you require.

During an RDQM queue manager failover the queue manager is ended, and its filesystem is unmounted on its current RDQM node. The filesystem is then mounted and the queue manager restarted on another node in the RDQM configuration. A filesystem cannot be unmounted if a process has an open handle for one of its files. To ensure a queue manager failover can complete, if the queue manager filesystem cannot be unmounted then processes that have an open file handle are sent a `SIGTERM` signal, followed by a `SIGKILL` if the open handles are not released. Your applications must be designed to respond correctly to `SIGTERM`. If applications or processes are configured as a queue manager service then, during a managed failover, they can be ended during queue manager shutdown before the filesystem is unmounted. If an application or process is not configured as a queue manager service, or an unmanaged failover occurs, such as a loss of quorum, then it is likely that signals will be sent to release the filesystem.

Managing primary and secondary characteristics of DR RDQMs

You can change a secondary disaster recovery replicated data queue manager (DR RDQM) into a primary DR RDQM. You can also change a primary instance into a secondary instance.

About this task

You use the **rdqmdr** command to change a secondary instance of an RDQM into the primary instance. You might need to complete this action if you lose your primary instance for some reason. You can then start the queue manager and carry on running it on the recovery node.

You also use the **rdqmdr** command to change a primary instance of an RDQM into the secondary instance. You might need to complete this action, for example, if you were reconfiguring your system.

You can also use the **rdqmdr** on a primary queue manager to retrieve the exact command that you need to create a secondary instance of that queue manager on your recovery node.

You can use the **rdqmdr** command as a user in the mqm group if the user can use sudo. Otherwise you must be logged in as root.

Procedure

- To change a secondary instance of a DR RDQM into a primary instance, enter the following command:

```
rdqmdr -m QMname -p
```

This command fails if the primary instance of the queue manager is still running and the DR replication link is still functioning.

- To change a primary instance of the queue manager into a secondary instance, enter the following command:

```
rdqmdr -m QMname -s
```

- To display the **crtmqm** command required to configure the secondary instance of a queue manager, enter the following command on your primary node:

```
rdqmdr -d -m QMname
```

You can enter the returned **crtmqm** command on your secondary node to create the secondary instance of the DR RDQM.

Starting, stopping, and displaying the state of a DR RDQM

You use variants of standard IBM MQ control commands to start, stop, and view the current state of a disaster recovery replicated data queue manager (DR RDQM).

About this task

You must run the commands that start, stop, and view the current state of a replicated data queue manager (RDQM) as a user that belongs to the mqm group.

You must run the commands to start and stop a queue manager on the primary node for that queue manager (that is, the node on which the queue manager is currently running).

Procedure

- To start a DR RDQM, enter the following command on the RDQM's primary node:

```
strmqm qmname
```

where *qmname* is the name of the RDQM that you want to start.

- To stop an RDQM, enter the following command on the RDQM's primary node:

```
endmqm qmname
```

where *qmname* is the name of the RDQM that you want to stop.

- To view the state of an RDQM, enter the following command:

```
dspmq -m QMname
```

The state information that is output depends on whether you run the command on the RDQM's primary or secondary node. If run on the primary node then one of the normal status messages returned by **dspmq** is displayed. If you run the command on a secondary node then the status Ended immediately is displayed. For example, if **dspmq** is run on node RDQM7, the following information might be returned:

QMNAME(DRQM8)	STATUS(Ended immediately)
QMNAME(DRQM7)	STATUS(Running)

You can use arguments with dspmq to establish whether an RDQM is configured for disaster recovery, and whether it is currently the primary or the secondary instance:

```
dspmq -m QMname -o (dr | DR)
```

One of the following responses is displayed:

DRROLE()

Indicates that the queue manager is not configured for disaster recovery.

DRROLE(Primary)

Indicates that the queue manager is configured as the DR primary.

DRROLE(Secondary)

Indicates that the queue manager is configured as the DR secondary.

Related reference

[dspmq](#)

[endmqm](#)

[strmqm](#)

Viewing DR RDQM status

You can view the status of all disaster recovery replicated data queue managers (DR RDQMs) on a node, or detailed information for a specified DR RDQM.

About this task

You use the **rdqmstatus** command to view the status of all DR RDQMs, or of individual RDQMs.

The summary status for a node also displays information about the DRBD kernel module that RDQM relies upon. When you upgrade RDQM, it is important to ensure that the correct version of the DRBD kernel module is installed for the version of RHEL kernel running on the system. The status displays the version of the OS kernel, the kernel version that the DRBD module was built for, the DRBD version, and the DRBD kernel module loaded status.

You must be a user in the mqm group to run the **rdqmstatus** command. You can run the command on either node of the DR RDQM pair.

Procedure

- To view the summary status of all the DR RDQMs on a node, run the following command on that node:

```
rdqmstatus
```

The status of the DR RDQMs on the node is displayed, for example:

```
Node: mqhvm07.exampleco.com
OS kernel version: 5.14.0-362.18.1
DRBD OS kernel version: 5.14.0-362.18.1
DRBD version: 9.2.7
DRBD kernel module status: Loaded

Queue manager name: DRQM8
Queue manager status: Ended immediately
DR role: Secondary

Queue manager name: DRQM7
Queue manager status: Running
DR role: Primary
```

The DRBD kernel module status is one of the following values:

Loaded

Indicates that the DRBD module has been loaded.

Partially loaded

Can occur when the DRBD module has been loaded, but does not function correctly due to a mismatch.

Not loaded

The DRBD module is not loaded. This can be displayed on a newly installed configuration, when no RDQM queue managers have yet been created.

Not installed

Indicates that either the DRBD module is not installed, or that IBM MQ was unable to determine the OS kernel version of the DRBD module.

Previously installed version still loaded

This status can arise if a new DRBD module is installed while the existing DRBD module is running (that is, an RDQM queue manager is running). The newly installed module is reported in the status, but is not the module that is actually running.

- To view the status of a particular RDQM, enter the following command:

```
rdqmstatus -m qmname
```

The following table summarizes the information that is returned.

<i>Table 33. Status attributes</i>		
Status Attribute	Possible Values	When Displayed
Queue manager status	state (as displayed by dspmq)	Always displayed
CPU	<i>n.nn%</i>	Only shown when RDQM on current node has primary role
Memory	<i>nnnMB</i>	Only shown when RDQM on current node has primary role
Queue manager file system	<i>nnnMB used, n.nGB allocated [n%]</i>	Only shown when RDQM on current node has primary role
DR role	Primary Secondary Unknown	Always displayed
DR Status	Normal	Normal operation
	Synchronization in progress	Synchronization is in progress

Table 33. Status attributes (continued)		
Status Attribute	Possible Values	When Displayed
	Partitioned	The queue manager has been started on both nodes while the DR replication network is unavailable
	Remote system unavailable	The connection to the other node has been lost
	Inconsistent	A synchronization was in progress, but was interrupted
	Reverting to snapshot	The user has chosen to revert to the snapshot that was taken when the queue manager entered the Inconsistent state.
	Remote system not configured	The primary instance of the RDQM has been configured, but no secondary instance has been configured
	Failed negotiation	One of the nodes has been set to synchronous replication and the other to asynchronous replication
DR type	Synchronous or asynchronous	Always displayed
DR port	<i>port_number</i> (the TCP/IP port used to replicate the data for this queue manager)	Always displayed
DR local IP address	The local IP address this queue manager is replicating from for DR	Always displayed
DR remote IP address	The remote IP address this queue manager is replicating to for DR	Always displayed
 DR TLS encryption	Yes No Not configured	Always displayed. Indicates whether the DR replication connection is configured to be secured with TLS, and if it is, whether the configuration is active.
DR out of sync data	<i>n</i> KB	Displayed when remote node unavailable or inconsistent
DR synchronization progress	<i>n</i> %	Displayed when synchronization is in progress
DR estimated time to completion	YYYY-MM-DD HH:MM:SS	Displayed when synchronization is in progress

Table 33. Status attributes (continued)		
Status Attribute	Possible Values	When Displayed
Snapshot reversion progress	n%	Displayed when DR status is Reverting to snapshot. The status counts down, so 0% shows completion
DR last in sync	YYYY-MM-DD HH:MM:SS	Displayed when DR data is out of sync (after initial synchronization). Gives the time and date when the data was last in sync.

Example

Example of normal status on primary node:

```
Queue manager status: Running
CPU: 0.00
Memory: 123MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
DR role: Primary
DR status: Normal
DR type: Synchronous
DR port: 3000
DR local IP address: 192.168.20.1
DR remote IP address: 192.168.20.2
DR TLS encryption: Yes
```

Example of normal status on a secondary node:

```
Queue manager status: Ended immediately
DR role: Secondary
DR status: Normal
DR port: 3000
DR local IP address: 192.168.20.2
DR remote IP address: 192.168.20.1
DR TLS encryption: Yes
```

Example of status on primary node when synchronization is in progress:

```
Queue manager status: Running
CPU: 0.53
Memory: 124MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
DR role: Primary
DR status: Synchronization in progress
DR type: Synchronous
DR port: 3000
DR local IP address: 192.168.20.1
DR remote IP address: 192.168.20.2
DR TLS encryption: Yes
DR synchronization progress: 11.0%
DR estimated time to completion: 2017-09-06 14:55:05
```

Example of a primary node, showing it is partitioned:

```
Queue manager status: Running
CPU: 0.02
Memory: 124MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
DR role: Primary
DR status: Partitioned
DR type: Synchronous
DR port: 3000
DR local IP address: 192.168.20.1
DR remote IP address: 192.168.20.2
DR TLS encryption: Yes
```

Example of a primary node, showing it is out of sync with the secondary node:

```
Queue manager status:      Running
CPU:                      0.00
Memory:                   123MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
DR role:                  Primary
DR status:                Remote unavailable
DR type:                  Asynchronous
DR port:                  3000
DR local IP address:      192.168.20.1
DR remote IP address:     192.168.20.2
DR TLS encryption:        Yes
DR out of sync data:      15932KB
DR last in sync:          2020-07-27 16:01:47
```

Example of a summary status showing a mismatch between the OS kernel version (RHEL 9.3) and the DRBD kernel module (targeted for RHEL 9.2). Even though the status reports that DRBD kernel module is loaded and the expected queue manager is running, you should update DRBD kernel module with the version targeted for the running OS kernel in this situation.

```
Node:                     mqhvm07.exampleco.com
OS kernel version:        5.14.0-362.18.1
DRBD OS kernel version:   5.14.0-284.11.1
DRBD version:             9.2.7+ptf.14
DRBD kernel module status: Loaded

Queue manager name:       DRQM8
Queue manager status:     Ended immediately
DR role:                  Secondary

Queue manager name:       DRQM7
Queue manager status:     Running
DR role:                  Primary
```

Example of a summary status showing a mismatch between the OS kernel version (RHEL 8.10) and the DRBD kernel module (targeted for RHEL 8.8). In this example the version mismatch is more severe and DRBD kernel module fails to load successfully. QM3 is a DR queue manager and is meant to be the Primary instance, but since the DRBD kernel module did not fully load, it reports as Secondary with a DR status of Unknown. To resolve this failure the DRBD kernel module must be updated with the version target for the running OS kernel.

```
Node:                     mqhvm57.exampleco.com
OS kernel version:        4.18.0-553
DRBD OS kernel version:   4.18.0-477
DRBD version:             9.2.7+ptf.14
DRBD kernel module status: Partially loaded

Queue manager name:       QM3
Queue manager status:     Status not available
DR role:                  Secondary
DR status:                Unknown
```

Related reference

 [rdqmstatus](#)

Operating in a disaster recovery environment

There are a number of situations in which you might want to switch over to the secondary queue manager in a disaster recovery configuration.

Disaster recovery

Following the complete loss of the primary queue manager at the main site, you start the secondary queue manager at the recovery site. Applications reconnect to the queue manager at the recovery site and the secondary queue manager processes application messages. The steps taken to revert to the previous configuration depend on the cause of the failure. For example, complete loss of main node versus temporary loss.

For steps to take following a temporary loss of the main site, see [“Switching over to a recovery node for a test scenario” on page 680](#). For steps to take following permanent failure, see [“Replacing a failed node in a disaster recovery configuration” on page 682](#).

Disaster recovery test support

You can test the disaster recovery configuration by temporarily switching over to the secondary instance and checking that applications can successfully connect. You follow the same procedure as when you switch over following a temporary failure of the primary node, see [“Switching over to a recovery node for a test scenario” on page 680](#).

Reverting to snapshot

If you suffer a failure in the primary node while a synchronization is in progress, you can revert to the snapshot taken of the secondary queue manager data just before the synchronization started. The secondary is then restored to a consistent state and can be run as the primary. To revert to the snapshot, you make the secondary into the primary, as described in [“Switching over to a recovery node for a test scenario” on page 680](#). You must check that the revert to snapshot has completed (by using the **rdqmdrstatus** command) before you start the queue manager.

Switching over to a recovery node for a test scenario

You can perform a manual fail over to test your RDQM disaster recovery (DR) configuration.

About this task

To test your RDQM DR configuration, you end the primary queue manager at the main site, then make secondary queue manager at the recovery site into the primary and start it. Applications reconnect to the queue manager at the recovery site and the queue manager processes application messages.

Important: You must ensure that the primary instance of a queue manager is stopped and made into a secondary instance before you promote the original secondary instance. Otherwise partitioned data can accrue.

You must either be logged in as root or logged in as a user who belongs to the mqm group and has the necessary sudo configuration.

Procedure

1. Stop the primary instance and re-designate it as the secondary instance:

```
endmqm qmname  
rdqmdr -m qmname -s
```

2. Make the secondary queue manager into the primary by entering the following command on the recovery node:

```
rdqmdr -m qmname -p
```

3. Start the queue manager by entering the following command:

```
strmqm qmname
```

4. Ensure that your applications reconnect to the queue manager on the recovery queue manager. Provided that you have defined your channels with a list of alternative connection names, specifying your primary and secondary queue managers, then your applications will automatically connect to the new primary queue manager.

What to do next

To return to normal operation, you must stop the queue manager on the recovery node, then promote the queue manager on the original node back to the primary role.

Related reference

[strmqm](#)

[rdqmdr](#)

If a disaster occurs in your main site, you take steps to switch over to your recovery site.

About this task

Following the loss of the primary queue manager at the main site, you make the secondary queue manager at the recovery site into the primary and start it. Applications reconnect to the queue manager at the recovery site and the queue manager processes application messages.

You should try to ensure that the primary queue manager at the main site is either unable to run, or has been stopped before you promote the original secondary instance. Otherwise partitioned data can accrue. Similarly, when the main site recovers, you should try to ensure that the queue manager does not restart and run as a primary instance. If this is not possible, block the replication traffic between the two instances until you can restore the correct primary/secondary relationship.

You must either be logged in as root or logged in as a user who belongs to the mqm group and has the necessary sudo configuration.

Procedure

1. On the recovery site, make the secondary queue manager into the primary by entering the following command on the recovery node:

```
rdqmdr -m qmname -p
```

2. Start the queue manager by entering the following command:

```
strmqm qmname
```

3. Ensure that your applications reconnect to the queue manager on the recovery queue manager. Provided that you have defined your channels with a list of alternative connection names, specifying your primary and secondary queue managers, then your applications will automatically connect to the new primary queue manager.

What to do next

When the failed node is restored, try to ensure that the queue manager does not restart and resume running. To return to normal operation, you must stop the queue manager on the recovery node, then promote the queue manager on the original node back to the primary role.

Related reference

[strmqm](#)

[rdqmdr](#)

Testing the recovery RDQM queue manager

You can test that the recovery instance of a queue manager in an RDQM disaster recovery configuration is operating correctly without disrupting the main site.

About this task

You test the recovery queue manager by disabling the interface between main and recovery nodes. You make the secondary queue manager into the primary and can then test the stand-alone queue manager. After testing is complete, you restore the interface and delete the test queue manager. You then re-create the queue manager as the secondary queue manager in the disaster recovery configuration.

Procedure

1. Disable the network connection between the main node and the recovery node.
2. On the recovery node, make the queue manager the primary:

```
rdqmdr -m QMname -p
```

Where *QMname* is the name of the queue manager.

3. Start the queue manager:

```
strmqm QMname
```

4. Connect applications to the queue manager and test that they work as expected.
5. End the queue manager:

```
endmqm QMname
```

6. Delete the queue manager:

```
dlrmqm QMname
```

7. Restore the network connection between the main and recovery appliances.
8. On the main node, run the following command to retrieve the **crtmqm** command that you used when you first configured disaster recovery.

```
rdqmdr -d -m QMname
```

9. Run the resulting **crtmqm** command on the recovery node to recreate the secondary queue manager. The primary queue manager on the main node synchronizes its data with the secondary queue manager to bring it up to date.

Replacing a failed node in a disaster recovery configuration

If you lose one of the nodes in a disaster recovery configuration, you can replace the node and restore the disaster recovery configuration by following this procedure.

About this task

If a disaster occurs such that the node in the main site is beyond repair, you can replace the failed node while the queue manager runs on the recovery node and then restore the original disaster recovery configuration. The replacement node must assume the identity of the failed node: the name and IP address must be the same.

You must either be logged in as root or logged in as a user who belongs to the mqm group and has the necessary sudo configuration.

Procedure

Following the loss of the queue manager on the main site, take the following steps:

1. On the recovery node, run the following commands to make the secondary queue manager assume the primary role:

```
rdqmdr -m QMname -p
```

Where *QMname* is the name of the queue manager.

2. Retrieve the command that you will need to run on the replacement primary node to reconfigure disaster recovery:

```
rdqmdr -m QMname -d
```

Copy the output of this command.

3. Run the following command to start the queue manager:

```
strmqm QMname
```

4. Ensure that your applications reconnect to the queue manager on the recovery node. Provided that you have defined your channels with a list of alternative connection names, specifying your primary and secondary queue managers, then your applications will automatically connect to the new primary queue manager.
5. Replace the failed node on your main site and configure it to have the same name and IP address that you used for disaster recovery on the original node. Then configure disaster recovery by running the **crtmqm** command that you copied in step 2. You now have a secondary instance of the queue manager, and the primary instance synchronizes its data with the secondary instance.
6. End the current primary instance.
7. After the synchronization has completed, make the primary instance that is running on the recovery node into the secondary once more:

```
rdqmdr -m QMname -s
```

8. On the replacement primary node, make the secondary instance of the queue manager into the primary instance:

```
rdqmdr -m QMname -p
```

9. On the replacement primary node, start the queue manager:

```
strmqm QMname
```

You have now restored the configuration as it was before the failure at your main site.

Related reference

[strmqm](#)

[rdqmdr](#)

[endmqm](#)

Resolving an inconsistent problem in DR RDQM

A DR status of *inconsistent* can be reported if synchronization fails between the primary and secondary instances of a queue manager.

About this task

An inconsistent state is reported on the secondary instance of a queue manager because the replication connection to the primary instance is lost during a synchronization operation. You might have to take action to resolve this situation. Consider the following sequence of events:

1. DR primary queue manager in sync with DR secondary queue manager
2. Replication link lost between primary and secondary
3. Replication link restored between primary and secondary
4. A resynchronization occurs where the DR secondary queue manager catches up with the DR primary queue manager. During this time the DR status of synchronization *in progress* is reported for both queue managers.
5. If the replication is then lost again during the resynchronization, the status on the DR secondary is reported as *Inconsistent*.

If the node hosting the primary queue manager is still operational, and the replication link can be restored, resynchronization happens automatically. The inconsistent state is resolved without you taking any action.

If the node hosting the primary queue manager is no longer operational, you can resolve the inconsistent state by implementing a revert to snapshot on the secondary queue manager. This operation reverts the data to the last known good state.

Procedure

To resolve an inconsistent state:

1. On the recovery node, make the secondary instance into the primary instance:

```
rdqmdr -m qmname -p
```

The revert to snapshot operation starts.

2. On the recovery node, check the status of the queue manager to see when the revert to snapshot operation is complete:

```
rdqmstatus -m qmname
```

3. When the queue manager status is Normal, start the queue manager:

```
strmqm qmname
```

Resolving a partitioned (split brain) problem in DR RDQM

A partitioned problem can occur if both queue managers in a disaster recovery pair run in the primary role at the same time.

About this task

If you promoted the secondary instance of a queue manager on the recovery node while the original primary instance continued to run on the main node, then you effectively have two versions of the same queue manager running, each with their own view of the queue manager data. The DR status for the queue manager on each node is reported as Partitioned.

You must decide which of the two queue managers has the most correct view of the data, and retain that set while discarding the other. You use the **rdqmdr** command to complete this operation.

There are two procedures. The first describes keeping the data from the main node, the second describes keeping the data from the recovery node.

Procedure

- To keep the data from the queue manager on the main node:
 - a) Ensure both queue manager instances are stopped.
 - b) Specify that the queue manager on the recovery node is the secondary:

```
rdqmdr -m qmname -s
```

- c) Specify that the queue manager on the main node is the primary:

```
rdqmdr -m qmname -p
```

Synchronization begins, with the data from the queue manager on the main node being copied to the recovery node.

- d) Check the status of the synchronization:

```
rdqmstatus -m qmname
```

- e) When the synchronization is complete, start the queue manager on the main node:

```
strmqm qmname
```

- To keep the data from the queue manager on the recovery node:
 - a) Ensure both queue manager instances are stopped.
 - b) Specify that the queue manager on the main node is the secondary:


```
rdqmdr -m qmname -s
```

- c) Specify that the queue manager on the recovery node is the primary:

```
rdqmdr -m qmname -p
```

Synchronization begins, with the data from the queue manager on the recovery node being copied to the main node.

- d) Check the status of the synchronization:

```
rdqmstatus -m qmname
```

- e) When the synchronization is complete, demote the queue manager on the recovery node:

```
rdqmdr -m qmname -s
```

- f) Promote the queue manager on the main node, and start it:

```
rdqmdr -m qmname -p  
strmqm qmname
```

Changing IP addresses in disaster recovery configurations

If you change the IP addresses of either of the interfaces in a disaster recovery configuration, replication is no longer possible between the two nodes.

If you need to change IP addresses for the replication interface for either of your DR nodes, you must use the following procedure:

1. On the primary node, back up the DR queue managers and then delete them. On the recovery node, delete the queue managers. see [“Backing up and restoring IBM MQ queue manager data” on page 759](#) and [“Deleting a DR RDQM” on page 667](#).
2. Recreate the DR queue managers, specifying the new IP addresses, and restore the back ups, see [“Creating a disaster recovery RDQM” on page 665](#) and [“Backing up and restoring IBM MQ queue manager data” on page 759](#).

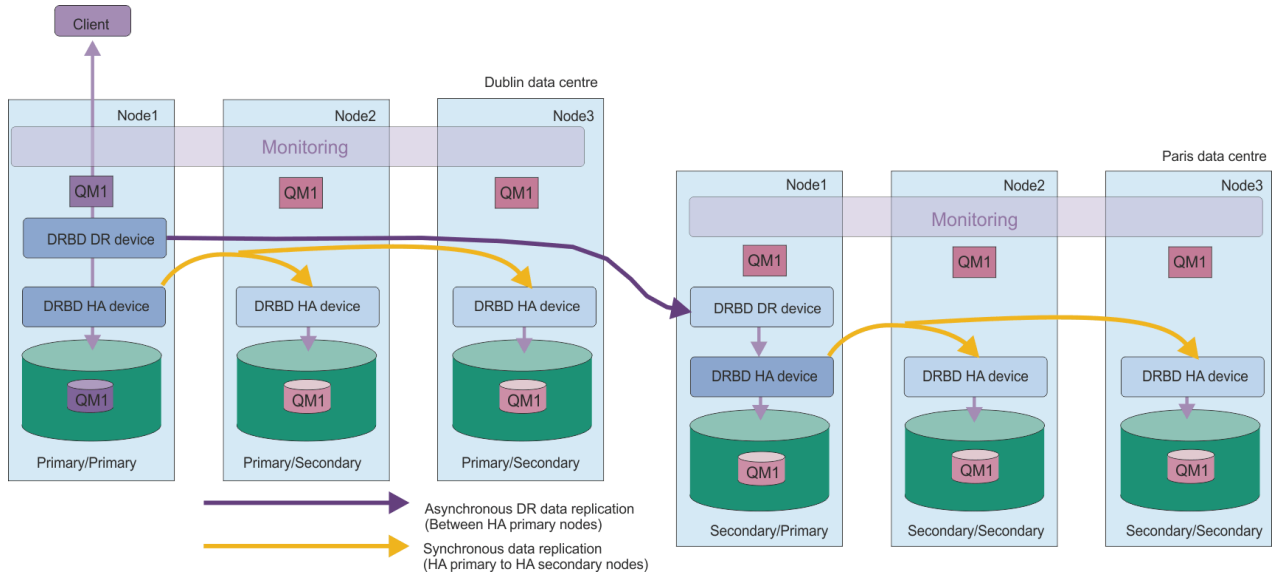
Linux

RDQM disaster recovery and high availability

You can configure a replicated data queue manager (RDQM) that runs on a high availability group on one site, but can fail over to another high availability group at another site if some disaster occurs that makes the first group unavailable. This is known as a DR/HA RDQM.

A DR/HA RDQM combines the features of a high availability RDQM (see [“RDQM high availability” on page 622](#)) and a disaster recovery RDQM (see [“RDQM disaster recovery” on page 658](#)).

The following diagram shows an example DR/HA RDQM.



The replication between the DR/HA RDQMs on the main site and the disaster recovery site is always asynchronous. With asynchronous replication, operations such as IBM MQ PUT or GET complete and return to the application before the event is replicated to the secondary queue manager.

You can have two active sites rather than 'main' and 'recovery' sites, if required, so some of your DR/HA RDQMs run on one site and some on the other during normal operation. If a disaster occurs and one site becomes unavailable, then all DR/HA RDQMs run on the same HA group at the same site.

Each HA group is configured in the same way as an ordinary HA group. You can define floating IP addresses for a DR/HA RDQM in each HA group. The floating IP address can be the same or different for each HA group.

You cannot upgrade an existing RDQM to be a DR/HA RDQM, you must create a DR/HA RDQM. (If required, you could back up the data of an existing RDQM, delete it, recreate it as a DR/HA RDQM, and then restore the data, see [“Backing up and restoring IBM MQ queue manager data”](#) on page 759.)

To configure DR/HA RDQMs, you must complete the following major steps:

1. Configure an HA group on the 'main' site.
2. Configure an HA group on the 'recovery' site.
3. Create a primary/primary DR/HA RDQM on one node of the HA group in the 'main' site.
4. Create primary/secondary DR/HA RDQMs on the other two nodes in the 'main' site.
5. Define a floating IP address for an application to access the DR/HA RDQM when it is running on any of the nodes of the HA group on the 'main' site.
6. Create a secondary/primary DR/HA RDQM on one node of the HA group on the 'recovery' site.
7. Create secondary/secondary DR/HA RDQMs on the other two nodes in the 'recovery' site.
8. Define a floating IP address for an application to access the DR/HA RDQM when it is running on any of the nodes of the HA group on the 'recovery' site.

Details about each of these steps are given in the following topics.

Requirements for a DR/HA RDQM solution

The requirements for the DR/HA RDQM solution are the same as for the HA RDQM solution and the DR RDQM solution.

For details of the requirements for the HA parts of the configuration, see [“Requirements for RDQM HA solution”](#) on page 625.

For details of the DR part of the configuration, see [“Requirements for RDQM DR solution”](#) on page 660.

Linux Configuring HA groups for DR/HA RDQMs

You must create an HA group on both your main and recovery sites. If you have an existing HA group on either site, you can create DR/HA RDQMs in that HA group. (Existing RDQMs will continue to operate as before.)

The procedure is the same as described for RDQM high availability, see [“Defining the Pacemaker cluster \(HA group\)”](#) on page 630.

When defining a high availability group you specify the IP addresses used for monitoring and replication by each node in the `rdqm.ini` file. When creating an HA group to support DR/HA RDQMs you can also specify the IP addresses used for DR replication by the HA group you are defining, and the IP addresses used for DR replication by the nodes in the other HA group of the DR pair. (If you do not specify the DR replication IP addresses in the `rdqm.ini` file, you can specify them in the command line when you create a DR/HA RDQM.)

If you are configuring an existing HA group, you can add DR replication IP addresses to your existing `rdqm.ini` file. You do not need to run `rdqmadm` again after updating `rdqm.ini`, but you must update `rdqm.ini` before you create any DR/HA RDQMs.

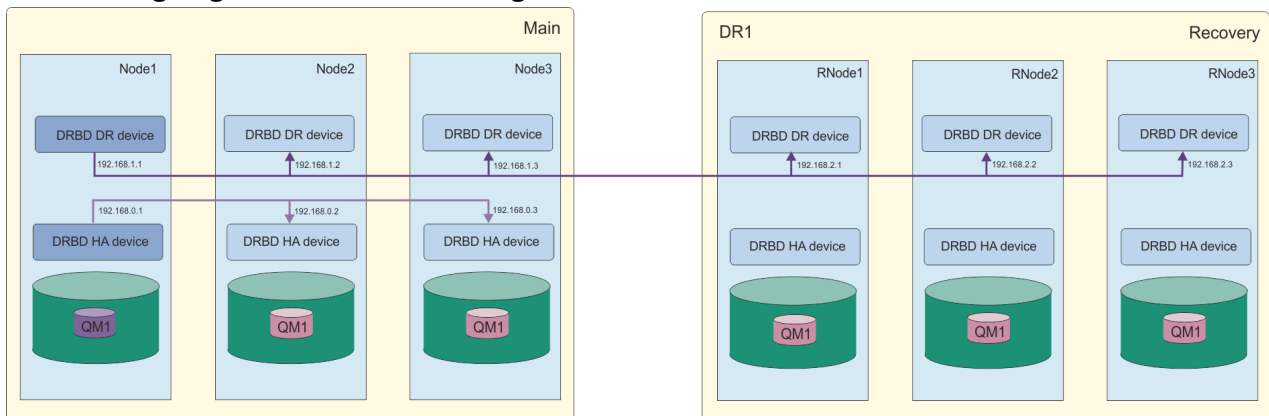
Use the `DR_Replication` attribute in the Node stanzas to specify the DR replication interfaces on the HA group that you are defining, for example:

```
Node:
  Name=Node1
  HA_Replication=192.168.0.1
  DR_Replication=192.168.1.1
Node:
  Name=Node2
  HA_Replication=192.168.0.2
  DR_Replication=192.168.1.2
Node:
  Name=Node3
  HA_Replication=192.168.0.3
  DR_Replication=192.168.1.3
```

Use the `DRGroup` stanza to specify the DR replication addresses of the remote HA group, for example:

```
DRGroup:
  Name=DR1
  DR_Replication=192.168.2.1
  DR_Replication=192.168.2.2
  DR_Replication=192.168.2.3
```

The following diagram illustrates this configuration:



If you do not specify DR replication IP addresses for the nodes in the local HA group in either the `rdqm.ini` file or on the command line when you create a DR/HA RDQM, then the `HA_Replication` interfaces defined for each node are used for DR replication. You must specify remote HA group DR replication addresses either in the `rdqm.ini` file or on the `crtmqm` command line.

Secure replication for DR/HA RDQM

You can specify that replication links in your DR/HA RDQM configuration are secured with TLS.

When you create a queue manager in a DR/HA RDQM configuration, you can specify that the links used to replicate data between the queue manager instances in the HA groups are secured with TLS. You can also specify that the link used to replicate data between the two HA configurations is secured with TLS. You can choose to secure either the HA or DR replication links, or both.

To configure secure replication, complete the following steps:

1. Configure the required certificates.
2. Configure the `tlshd` service.
3. Specify the `-re`, `-reh`, or `-red` option when creating a new queue manager using **crtmqm**.

Complete the first two steps once per node. The certificates and `tlshd` configuration are shared by all RDQM queue managers on the node that are configured to use secure replication links.

The TLS protocols and cipher suites that can be used by the RDQM secure replication links are controlled by the active system-wide cryptographic policy. Use the Linux **update-crypto-policies** command to change the current active policy.

Configuring certificates

You must configure the certificates to be used by the replication link. The certificates that you create for each node can be used by any replication links. If you have more than one RDQM queue manager on a node that uses secure replication links, the same certificate is used.

To use secure replication links in a RDQM DR/HA configuration you need six certificates, one certificate per node in the RDQM DR/HA group.

To configure your certificates complete the following steps for each node in your RDQM configuration:

1. Use the following command to generate a private key:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -out key_file
```

where *key_file* is the name of the file that is created to store the private key.

2. Create a certificate signing request.

In a RDQM DR/HA configuration, the certificate Subject Alternative Name (SAN) must contain both the hostname of the RDQM node and a group DNS name. The group DNS name defaults to the value `encrypted.remote`. If required, you can specify a different group DNS name.

The same group DNS name can be used by both HA groups, or you can specify a different name for each HA group.

Every node in a HA group must use certificates with the same group DNS name in the SAN.

If you do not use the default value of `encrypted.remote` in the SAN when the certificate is created, you must specify the group DNS name for the local and remote HA groups by using the **-san** parameter on the **crtmqm** command when the queue manager is created.

Use the following command to create a certificate signing request:

```
openssl req -key key_file -subj "/CN=hostname"
           -addext "subjectAltName = DNS:hostname,DNS:group_dns_name"
           -new -out cert_req_file
```

where:

- *key_file* is the file that contains the private key that you created in step “1” on page 688.
- *hostname* is the hostname of the node.
- *group_dns_name* is the group DNS name. To use the default value for the group DNS name, specify `encrypted.remote`.

- *cert_req_file* is the file that is created to store the certificate signing request.
3. Send the certificate signing request to your certificate authority (CA) to be signed. When you receive the signed certificate from the CA, copy the private key, signed certificate, and CA certificate files to the directory where you want to store these files. For example, the `/etc/drbd.d/` directory.
 4. Set the file permissions and ownership of the private key and certificate files to ensure that only the root user can view the private keys.
 - Change the ownership of the private key and certificate file to the root user.
 - Set the permissions on the private key file so that only the root user has read and write access.

The following example script creates an internal certificate authority and three certificates for use in a RDQM DR/HA configuration.

Important: This example is not suitable for a production environment, and is intended only as an example to set up a test environment quickly. Certificate management is a complex subject for advanced users. For production, you must consider things like rotation, revocation, key length, and much more.

Run the following script on one of the nodes in your primary HA group:

```
#!/bin/bash

# The CA name
ca_name="RDQM Test CA"

# The CA filename
ca_filename="rdqm_test_ca"

# Host certificates to generate
test_hosts=("alice" "bob" "charlie")

mkdir certificates && cd ./certificates

# Create private key for use with private CA using OpenSSL
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 \
-out $ca_filename.key

# Create a private CA certificate using OpenSSL
openssl req -key $ca_filename.key -new -x509 -days 3650 -subj "/CN=$ca_name" \
-out $ca_filename.crt

# Create certificates for each node
for host in ${test_hosts[@]}; do

    # Create a private key
    openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 \
    -out $host.key

    # Generate a CSR
    openssl req -key $host.key -subj "/CN=$host" \
    -addext "subjectAltName = DNS:$host,DNS:encrypted.remote" \
    -new -out $host.csr

    # Create a valid certificate signed by the private CA
    openssl x509 -req -in $host.csr -CA $ca_filename.crt \
    -copy_extensions copyall -CAkey $ca_filename.key -CAcreateserial \
    -days 365 -out $host.crt

    # Remove sequential serial number
    rm -f $ca_filename.srl

    # Remove CSR
    rm -f $host.csr

done
```

Run the same script on one of the nodes in your secondary HA group, specifying the three hostnames of the nodes in that group.

The example generates the following files in the `certificates` directory on the node in the primary group:

- `rdqm_test_ca.crt`

- `rdqm_test_ca.key`
- `alice.crt`
- `alice.csr`
- `alice.key`
- `bob.crt`
- `bob.csr`
- `bob.key`
- `charlie.crt`
- `charlie.csr`
- `charlie.key`

If the hosts in your secondary group were called `william`, `joey`, and `kayser`, then the following files are generated in the `certificates` directory on the node in the secondary group:

- `rdqm_test_ca.crt`
- `rdqm_test_ca.key`
- `william.crt`
- `william.csr`
- `william.key`
- `joey.crt`
- `joey.csr`
- `joey.key`
- `kayser.crt`
- `kayser.csr`
- `kayser.key`

Deploy the certificates by completing the following steps:

1. Copy the CA certificate (`rdqm_test_ca.crt`) to each node's `/etc/drbd.d/` directory. (These are example paths, you can choose another location if required.)
2. Copy each node's certificate (`.crt` file) and private key (`.key` file) to `/etc/drbd.d/`. (These are example paths, you can choose another location if required.)
3. Change file permissions and ownership for the private keys and certificate on each node. This ensures that only the root user can view the contents of the private keys:

```
chmod 600 /etc/drbd.d/*.key
chown root:root /etc/drbd.d/*.key
chown root:root /etc/drbd.d/*.crt
```

4. Store the certificate authority's private key in a secure location.

Configuring the `tlshd` service

This `tlshd` service needs to be configured and then enabled and started, before any DRBD TLS handshakes can be completed. The `tlshd` service is provided in a package that is installed with the `drbd-utils-9` packages.

The configuration file for this service is `/etc/tlshd.conf` and the initial contents are:

```
[debug]
loglevel=0
tls=0
nl=0

[authenticate]
#keyrings= <keyring>;<keyring>;<keyring>
```

```
[authenticate.client]
#x509.truststore= <pathname>
#x509.certificate= <pathname>
#x509.private_key= <pathname>

[authenticate.server]
#x509.truststore= <pathname>
#x509.certificate= <pathname>
#x509.private_key= <pathname>
```

Before enabling and starting the `tlshd` service, you must specify values for the following properties in both the `authenticate.client` and `authenticate.server` sections (DRBD can establish its connections in either direction, unless there is a firewall blocking one direction):

x509.truststore

The path to the file that contains the certificate of the CA that signed the certificates used by the RDQM nodes. If a complex trust chain is required, it must be captured in a single file.

x509.certificate

The path to the file that contains the CA-signed certificate for the RDQM node.

x509.private_key

The path to the file that contains the private key for the RDQM node. The file must be owned by root and read/write permissions must be granted to only the root user.

The paths that are specified in these properties must all identify local files.

You can also optionally specify a certificate revocation list (CRL). You must set this in both the `authenticate.client` and `authenticate.server` sections:

x509.crl

The path to a file containing PEM-encoded certificate revocation lists (CRL) that are to be used to verify the revocation status of certificates during each handshake. If this option is not specified, CRL checking is skipped.

If you created the certificates using the example script in [“Configuring certificates” on page 688](#), then you would configure the file as follows for server alice:

```
[authenticate.client]
x509.truststore=/etc/drbd.d/rdqm_test_ca.crt
x509.certificate=/etc/drbd.d/alice.crt
x509.private_key=/etc/drbd.d/alice.key

[authenticate.server]
x509.truststore=/etc/drbd.d/rdqm_test_ca.crt
x509.certificate=/etc/drbd.d/alice.crt
x509.private_key=/etc/drbd.d/alice.key
```

Before you enable the service for the first time, enter the following command:

```
systemctl daemon-reload
```

To start the `tlshd` service, enter the following command:

```
systemctl enable --now tlshd
```

Creating the DR/HA RDQM

You create a DR/HA queue manager with secure HA and DR replication links by using the **crtmqm** command with the `-re` option, for example, for the primary instance in the primary group:

```
crtmqm -sx -rr p -rl <Local_IPs> -ri <Recovery_IPs> -rp <DRPort> -re <DRHAQueueManagerName>
```

You run the following command on each of the other nodes in the primary group to create the secondary instances:

```
crtmqm -sxs -rr p -rl <Local_IPs> -ri <Recovery_IPs> -rp <DRPort> -re <DRHAQueueManagerName>
```

You create the primary instance in the secondary group by using the following command:

```
crtmqm -sx -rr s -rl <Local_IPs> -ri <Recovery_IPs> -rp <DRPort> -re <DRHAQueueManagerName>
```

You run the following command on each of the other nodes in the secondary group to create the secondary instances:

```
crtmqm -sxs -rr s -rl <Local_IPs> -ri <Recovery_IPs> -rp <DRPort> -re <DRHAQueueManagerName>
```

To secure only the HA replication links in the group, specify the `-reh` option in the commands. For example:

```
crtmqm -sx -rr p -rl <Local_IPs> -ri <Recovery_IPs> -rp <DRPort> -reh <DRHAQueueManagerName>
```

To secure only the DR replication link between the groups, specify the `-red` option in the commands. For example:

```
crtmqm -sx -rr p -rl <Local_IPs> -ri <Recovery_IPs> -rp <DRPort> -red <DRHAQueueManagerName>
```

The previous examples assume that, where a secure DR replication link is required, you have created certificates for each of the nodes using the default value of `encrypted.remote` for the group DNS name in the certificate's SAN. Where you have created certificates using a different value, you must specify this value when you create the queue manager by using the `-san` option. For example:

```
crtmqm -sx -rr p -rl <Local_IPs> -ri <Recovery_IPs> -rp <DRPort> -red -san  
<customSubjectAltName> <DRHAQueueManagerName>
```

If you have used the same group DNS name in the certificates for both groups, then specify the value by using the `-san <customSubjectAltName>` option in all the **crtmqm** commands that create the DR/HA RDQM configuration.

If you have used a different group DNS name in the certificates for each HA group, then specify the group DNS names in the certificates of both the primary and secondary HA groups as comma-separated values, when creating the queue manager in each group. The following command creates the primary queue manager in the primary HA group:

```
crtmqm -sx -rr p -rl <Local_IPs> -ri <Recovery_IPs> -rp <DRPort> -red -san  
<primarySubjectAltName>,<secondarySubjectAltName> <DRHAQueueManagerName>
```

The following command creates the primary queue manager on the secondary HA group:

```
crtmqm -sx -rr s -rl <Local_IPs> -ri <Recovery_IPs> -rp <DRPort> -red -san  
<secondarySubjectAltName>,<primarySubjectAltName> <DRHAQueueManagerName>
```

(You can specify either the `-re` or `-red` option in these examples.)

Use the **rdqmstatus** command to confirm that the replication links are secured with TLS. For more information, see [“Viewing DR/HA RDQM and HA group status” on page 700](#).

For more information about creating DR/HA RDQMs, see [“Creating DR/HA RDQMs” on page 692](#).

Creating DR/HA RDQMs

You use the **crtmqm** command to create a replicated data queue manager (RDQM) in a DR/HA configuration.

About this task

You can create a DR/HA RDQM as a user in the mqm group if the user can use `sudo`. Otherwise you must create the RDQM as root.

You must create a number of DR/HA RDQMs:

- On the HA group on the 'main' site:

- On the node where you want the queue manager to run under normal conditions, create the primary/primary DR/HA RDQM.
- On each of the other two nodes in the HA group, create a primary/secondary DR/HA RDQM.
- On the HA group on the 'recovery' site:
 - On the node where the queue manager will run if it fails over to the recovery site, create the secondary/primary DR/HA RDQM. You can use the command output when you created the primary/primary queue manager on the 'main' site.
 - On each of the other two nodes in the HA group, create a secondary/secondary DR/HA RDQM.

All of the queue manager instances must have the same name and be allocated the same amount of storage.

The following points give some guidance about sizing the queue manager file system:

1. When creating an RDQM queue manager, a file system is allocated to store queue manager data and logs. It is important to size this file system appropriately so that the queue manager can record ongoing activity in its logs, and store application messages on queues. When sizing the file system, consider peak messaging requirements, future workload growth, and application outages that might cause messages to build up on queues. For guidance on calculating the size of the queue manager recovery log, see [“How large should I make my log filesystem?”](#) on page 737. When calculating the storage requirements for application messages, the size and number of messages needs to be considered, plus their MQMD header and any message properties they have.
2. RDQM queue manager file systems cannot be dynamically resized. You must back up then restore an RDQM queue manager with a larger file system if this is required, see [“Resizing the filesystem for an HA RDQM queue manager”](#) on page 643.
3. You can limit the size of individual queues on disk by using local queue attributes, such as MAXDEPTH and MAXFSIZE. See [Modifying IBM MQ queue files](#).
4. You should monitor your ongoing disk usage and respond appropriately if disk usage increases before file system usage becomes critical. File system usage can be monitored by using either platform/operating system capabilities, or by subscribing to metrics published to IBM MQ system topics that are described at [Metrics published on the system topics](#).

V10.0.0 You can, if required, specify that the replication links between the instances of the HA RDQMs within the primary and secondary groups are secured with TLS. You can also independently specify that the replication link between the primary and secondary groups is secured with TLS. If secure replication links are specified when a primary instance is created, then the secondary instances must be created with the same settings. If the replication link between the primary and secondary groups is secured with TLS, then this setting must be made on the primary and secondary instances in both the primary and secondary groups.

V10.0.0 To use secure replication links, you must configure the certificates and tlshd service before the DR/HA RDQM is created. For more information, see [“Secure replication for DR/HA RDQM”](#) on page 688.

Procedure

- To create the primary/primary DR/HA RDQM:
 - a) Enter the following command:

```
crtmqm -sx -rr p
      [-rl DRLocalIP1,DRLocalIP2,DRLocalIP3]
      (-ri DRRemoteIP1,DRRemoteIP2,DRRemoteIP3 | -in GroupName)
      -ip DRPort
      [((-re|-red) [-san SubjectAltName])|-reh]
      [-z] [-q] [-c Text] [-d DefXmitQ] [-h MaxHandles]
      [-g ApplicationGroup] [-oa user|group]
      [-t TrigInt] [-u DeadQ] [-x MaxUMsgs]
      [-lp LogPri] [-ls LogSec]
      [-lc | -ll | -lla | -lln] [-lf LogFileSize]
      [-p Port] [-fs FilesystemSize] QMgrName
```

Where:

-sx

Indicates that the initial HA role is primary.

-rr p

Indicates that the initial DR role is primary.

-rl *DRLocalIP1, DRLocalIP2, DRLocalIP3*

Optionally specify the IP addresses of the DR interfaces on the three nodes on the local site (that is, the 'main' site). If not specified, the IP addresses specified in the `rdqm.ini` file are used.

-ri *DRRemoteIP1, DRRemoteIP2, DRRemoteIP3*

Specify IP addresses of the DR interfaces on the three nodes on the remote site (that is, the 'recovery' site). You must specify either this or the `-rn` parameter.

-rn *GroupName*

Specify the remote HA group name as specified in the `rdqm.ini` file. You must specify either `-ri` or `-rn`.

-rp *Port*

Specifies the port to use for DR replication.

V 5000 -re [-san *SubjectAltName*]

Specify that both HA and DR replication links are secured with TLS.

The Subject Alternative Name (SAN) of the certificates that are configured for each HA group must contain both the hostname of the RDQM node and a group DNS name. The group DNS name defaults to the value `encrypted.remote`. Use the `-san` option to specify a different group DNS name, if the certificates are created with a SAN that contains a group DNS name other than the default value.

V 5000 -red [-san *SubjectAltName*]

Specify that only the DR replication connection is secured with TLS.

The Subject Alternative Name (SAN) of the certificates that are configured for each HA group must contain both the hostname of the RDQM node and a group DNS name. The group DNS name defaults to the value `encrypted.remote`. Use the `-san` option to specify a different group DNS name, if the certificates are created with a SAN that contains a group DNS name other than the default value.

V 5000 -reh

Specify that only the HA replication links are secured with TLS.

other_crtmqm_options

You can optionally specify one or more of these general **crtmqm** options:

- `-z`
- `-q`
- `-c Text`
- `-d DefaultTransmissionQueue`
- `-h MaxHandles`
- `-g ApplicationGroup`
- `-oa user | group`
- `-t TrigInt`
- `-u DeadQ`
- `-x MaxUMsgs`
- `-lp LogPri`
- `-ls LogSec`
- `-lc | -l`
- `-lla | -lln`
- `-lf LogFileSize`

– -p *Port*

For more information about these parameters, see [crtmqm \(create queue manager\)](#).

-fs size

Optionally specifies the size of the filesystem to create for the queue manager, that is, the size of the logical volume that is created in the drbdpool volume group. Another logical volume of that size is also created, to support the reverting to snapshot operation, so the total storage for the DR RDQM is just over twice that specified here.

Size is a numeric value, which is specified in GB. You can specify a value in MB by entering the value followed by the character M. For example, to specify a filesystem size of 3 GB, enter 3. To specify a filesystem size of 1024 MB, enter 1024M. (You can also add a G suffix to explicitly state GB.)

QMname

Specifies the name of the replicated data queue manager. The name is case sensitive.

After the command completes, it outputs the command that you can input on the recovery site to create the secondary/primary instance of the queue manager.

- To create a primary/secondary DR/HA RDQM on the other two nodes in the HA group:
 - a) Enter the following command on each node:

```
crtmqm -sxs -rr p
        [-rl DRLocalIP1,DRLocalIP2,DRLocalIP3]
        (-ri DRRemoteIP1,DRRemoteIP2,DRRemoteIP3 | -rn GroupName)
        -rp DRPort
        [((-re|-red) [-san SubjectAltName])|-reh]
        [-fs FilesystemSize] QMgrName
```

Where:

-sxs

Indicates that the initial HA role is secondary.

-rr p

Indicates that the initial DR role is primary.

-rl DRLocalIP1, DRLocalIP2, DRLocalIP3

Optionally specify the IP addresses of the DR interfaces on the three nodes on the local site (that is, the 'main' site). If not specified, the IP addresses specified in the `rdqm.ini` file are used.

-ri DRRemoteIP1, DRRemoteIP2, DRRemoteIP3

Specify IP addresses of the DR interfaces on the three nodes on the remote site (that is, the 'recovery' site). You must specify either this or the `-rn` parameter.

-rn GroupName

Specify the remote HA group name as specified in the `rdqm.ini` file. You must specify either `-ri` or `-rn`.

-rp Port

Specifies the port to use for DR replication.

-re [-san SubjectAltName]

Specify that both HA and DR replication connections are secured with TLS.

The Subject Alternative Name (SAN) of the certificates that are configured for each HA group must contain both the hostname of the RDQM node and a group DNS name. The group DNS name defaults to the value `encrypted.remote`. Use the `-san` option to specify a different group DNS name, if the certificates are created with a SAN that contains a group DNS name other than the default value.

-red [-san SubjectAltName]

Specify that only the DR replication connection is secured with TLS.

The Subject Alternative Name (SAN) of the certificates that are configured for each HA group must contain both the hostname of the RDQM node and a group DNS name. The group DNS

name defaults to the value `encrypted.remote`. Use the `-san` option to specify a different group DNS name, if the certificates are created with a SAN that contains a group DNS name other than the default value.

V 10.0.0 -reh

Specify that only the HA replication connections are secured with TLS.

-fs size

Specifies the size of the filesystem to create for the queue manager, that is, the size of the logical volume that is created in the drbdpool volume group. If you have specified a non-default size when creating the primary/primary RDQM, you must specify the same value here.

Size is a numeric value, which is specified in GB. You can specify a value in MB by entering the value followed by the character M. For example, to specify a filesystem size of 3 GB, enter 3. To specify a filesystem size of 1024 MB, enter 1024M. (You can also add a G suffix to explicitly state GB.)

QMname

Specifies the name of the primary/secondary RDQM. This must be the same as the name you specified for the primary/primary instance of the RDQM. Note that the name is case sensitive.

- To create a secondary/primary DR/HA RDQM on the node where the queue manager will run if it fails over to the recovery site:
 - a) Use the command output when you created the primary/primary DR/HA on the main site, or enter the following command:

```
crtmqm -sx -rr s
      [-rl DRLocalIP1,DRLocalIP2,DRLocalIP3]
      (-ri DRRemoteIP1,DRRemoteIP2,DRRemoteIP3 | -rn GroupName)
      -rp DRPort
      [((-re|-red) [-san SubjectAltName])|-reh]
      [-fs FilesystemSize] QMgrName
```

-sx

Indicates that the initial HA role is primary.

-rr s

Indicates that the initial DR role is secondary.

-rl DRLocalIP1, DRLocalIP2, DRLocalIP3

Optionally specify the IP addresses of the DR interfaces on the three nodes on the local site (that is, the 'recovery' site). If not specified, the IP addresses specified in the `rdqm.ini` file are used.

-ri DRRemoteIP1, DRRemoteIP2, DRRemoteIP3

Specify IP addresses of the DR interfaces on the three nodes on the remote site (that is, the 'main' site). You must specify either this or the `-rn` parameter.

-rn GroupName

Specify the remote HA group name as specified in the `rdqm.ini` file. You must specify either `-ri` or `-rn`.

-rp Port

Specifies the port to use for DR replication.

V 10.0.0 -re [-san SubjectAltName]

Specify that both HA and DR replication connections are secured with TLS.

The Subject Alternative Name (SAN) of the certificates that are configured for each HA group must contain both the hostname of the RDQM node and a group DNS name. The group DNS name defaults to the value `encrypted.remote`. Use the `-san` option to specify a different group DNS name, if the certificates are created with a SAN that contains a group DNS name other than the default value.

V 10.0.0 -red [-san SubjectAltName]

Specify that only the DR replication connection is secured with TLS.

The Subject Alternative Name (SAN) of the certificates that are configured for each HA group must contain both the hostname of the RDQM node and a group DNS name. The group DNS name defaults to the value `encrypted.remote`. Use the `-san` option to specify a different group DNS name, if the certificates are created with a SAN that contains a group DNS name other than the default value.

V 10.0.0 -reh

Specify that only the HA replication connections are secured with TLS.

-fs size

Optionally specifies the size of the filesystem to create for the queue manager, that is, the size of the logical volume that is created in the `drbdpool` volume group. Another logical volume of that size is also created, to support the reverting to snapshot operation, so the total storage for the DR RDQM is just over twice that specified here.

QMname

Specifies the name of the replicated data queue manager. The name is case sensitive.

- To create a secondary/secondary HA/DR RDQM on the other two nodes on the recovery site:
 - a) Enter the following command on each node:

```
critmqm -sxs -rr s
      [-rl DRLocalIP1,DRLocalIP2,DRLocalIP3]
      (-ri DRRemoteIP1,DRRemoteIP2,DRRemoteIP3 | -rn GroupName)
      -rp DRPort
      [((-re|-red) [-san SubjectAltName])|-reh]
      [-fs FilesystemSize] QMgrName
```

-sxs

Indicates that the initial HA role is primary.

-rr s

Indicates that the initial DR role is secondary.

-rl DRLocalIP1, DRLocalIP2, DRLocalIP3

Optionally specify the IP addresses of the DR interfaces on the three nodes on the local site. If not specified, the IP addresses specified in the `rdqm.ini` file are used.

-ri DRRemoteIP1, DRRemoteIP2, DRRemoteIP3

Specify IP addresses of the DR interfaces on the three nodes on the remote site. You must specify either this or the `-rn` parameter.

-rn GroupName

Specify the remote HA group name as specified in the `rdqm.ini` file. You must specify either `-ri` or `-rn`.

-rp Port

Specifies the port to use for DR replication.

V 10.0.0 -re [-san SubjectAltName]

Specify that both HA and DR replication connections are secured with TLS.

The Subject Alternative Name (SAN) of the certificates that are configured for each HA group must contain both the hostname of the RDQM node and a group DNS name. The group DNS name defaults to the value `encrypted.remote`. Use the `-san` option to specify a different group DNS name, if the certificates are created with a SAN that contains a group DNS name other than the default value.

V 10.0.0 -red [-san SubjectAltName]

Specify that only the DR replication connection is secured with TLS.

The Subject Alternative Name (SAN) of the certificates that are configured for each HA group must contain both the hostname of the RDQM node and a group DNS name. The group DNS name defaults to the value `encrypted.remote`. Use the `-san` option to specify a different group DNS name, if the certificates are created with a SAN that contains a group DNS name other than the default value.

-reh

Specify that only the HA replication connections are secured with TLS.

-fs size

Optionally specifies the size of the filesystem to create for the queue manager, that is, the size of the logical volume that is created in the drbdpool volume group. Another logical volume of that size is also created, to support the reverting to snapshot operation, so the total storage for the DR RDQM is just over twice that specified here.

QMname

Specifies the name of the replicated data queue manager. The name is case sensitive.

Note: When you create an RDQM, the next free port number above 7000 is allocated for the HA replication link. If it is discovered that the chosen port is used by another application, the **crtmqm** command fails with the AMQ6543 error and that port is added to an exclude list. You must delete the secondary instances of the queue manager, then run the **crtmqm** command again.

What to do next

After you have created all the DR/HA RDQMs, you must check the status on the primary/primary and secondary/primary instances to check all are correct. Use the **rdqmstatus** command on the nodes. The nodes should be displaying normal status as described in [“Viewing DR/HA RDQM and HA group status” on page 700](#). If they are not displaying this status, delete the secondary/primary instance and recreate it, taking care to use the correct arguments.

Related tasks

[“Creating DR/HA RDQMs” on page 692](#)

You use the **crtmqm** command to create a replicated data queue manager (RDQM) in a DR/HA configuration.

Related reference

[crtmqm](#)

Deleting a DR/HA RDQM

You use the **dltmqm** command to delete a DR/HA replicated data queue manager (RDQM).

About this task

You must run the command to delete the RDQM on both the primary/primary node and the secondary/primary node. The RDQM must be ended first. You can run the command as an mqm user if that user has the necessary sudo privileges. Otherwise, you must run the command as root.

Procedure

- To delete a DR/HA RDQM, enter the following command:

```
dltmqm RDQM_name
```

Related reference

[dltmqm](#)

Creating a floating IP address

You can create floating IP addresses for each of your HA groups in a DR/HA RDQM configuration.

A floating IP address enables a client to use the same IP address for a DR/HA RDQM regardless of which node in an HA group it is running on. If your two HA groups have private/isolated networks for application connectivity, then the same floating IP address can be defined for both groups. You must still define that floating IP address twice, however, once on each of your HA groups.

You create and delete floating IP addresses using the same method as for an HA RDQM. See [“Creating and deleting a floating IP address” on page 646](#).

You use variants of standard IBM MQ control commands to start, stop, and view the current state of a DR/HA RDQM.

About this task

You must run the commands that start, stop, and view the current state of a DR/HA RDQM as a user that belongs to both the `mqm` and `haclient` groups.

You must run the commands to start and stop a queue manager on the primary node for that queue manager.

Procedure

- To start an RDQM, enter the following command on the RDQM's primary node:

```
strmqm qmname
```

where *qmname* is the name of the DR/HA RDQM that you want to start.

The RDQM is started, and Pacemaker starts managing the RDQM. You must specify the `-ns` option with `strmqm` if you want to specify any other `strmqm` options.

- To stop an RDQM, enter the following command on the DR/HA RDQM's primary node:

```
endmqm qmname
```

where *qmname* is the name of the RDQM that you want to stop.

Pacemaker ceases to manage the RDQM, and then the RDQM is ended. All other **endmqm** parameters can be used when stopping an RDQM.

- To view the state of an RDQM, enter the following command:

```
dspmq -m QMname
```

The state information that is output depends on whether you run the command on the RDQM's primary or secondary node. If run on the primary node then one of the normal status messages returned by **dspmq** is displayed. If you run the command on a secondary node then the status **Ended immediately** is displayed. For example, if **dspmq** is run on node RDQM7, the following information might be returned:

QMNAME(DRQM8)	STATUS(Ended immediately)
QMNAME(DRQM7)	STATUS(Running)

You can use arguments with `dspmq` to establish whether an RDQM is configured for disaster recovery, and whether it is currently the primary or the secondary instance:

```
dspmq -m QMname -o (dr | DR)
```

One of the following responses is displayed:

DRROLE()

Indicates that the queue manager is not configured for disaster recovery.

DRROLE(Primary)

Indicates that the queue manager is configured as the DR primary.

DRROLE(Secondary)

Indicates that the queue manager is configured as the DR secondary.

Use the **dspmq -o all** command to view the disaster recovery and high availability information for DR/HA RDQMs. For example, if you run **dspmq -o all** on the node where the DR/HA RDQM is running, you see the following state information:

```
QMNAME (TESTQM1)
DRROLE (Primary)
```

```
STATUS (Running) HA (Replicated)
```

Related reference

[dspmq \(display queue managers\)](#)

[endmqm \(end queue manager\)](#)

[strmqm \(start queue manager\)](#)

Failed resource actions in DR/HA configurations

Failed resource actions arise when the Pacemaker component of an RDQM high availability configuration encounters some problem with a resource on one of the nodes in an HA group.

Failed resource actions might arise on either of the HA configurations in an RDQM DR/HA configuration. You can use the **rdqmstatus** command to view failed resource actions, and the **rdqmclean** command to clear them (after the cause of the failure has been resolved). The process is the same as for RDQM HA configurations without the DR component. See [“Failed resource actions” on page 649](#) for more details.

Related tasks

[“Viewing DR/HA RDQM and HA group status” on page 700](#)

You can view the HA status and DR role of DR/HA replicated data queue managers (RDQMs).

[“Viewing HA RDQM and HA group status” on page 650](#)

You can view the status of the HA group and of individual HA replicated data queue managers (RDQMs).

Related reference

[rdqmclean](#)

[rdqmstatus](#)

Viewing DR/HA RDQM and HA group status

You can view the HA status and DR role of DR/HA replicated data queue managers (RDQMs).

About this task

You use the **rdqmstatus** command to view the status of individual RDQMs or get an overview of the status of all the RDQMs known to the HA group.

The summary status for a node also displays information about the DRBD kernel module that RDQM relies upon. When you upgrade RDQM, it is important to ensure that the correct version of the DRBD kernel module is installed for the version of RHEL kernel running on the system. The status displays the version of the OS kernel, the kernel version that the DRBD module was built for, the DRBD version, and the DRBD kernel module loaded status.

Note: Note that, in an HA/DR configuration, the DR configuration always uses asynchronous replication, while the HA configuration always uses synchronous replication. These values are not displayed in the output of the **rdqmstatus -m qmgr** command in a combined HA/DR configuration.

You must be a user in the **mqm** and **haclient** groups to run the **rdqmstatus** command. You can run the command on any of the nodes in either of the HA groups.

Procedure

- To view the summary status of a node and the RDQMs that are part of the HA configuration:

```
rdqmstatus
```


The identity of the node that you ran the command on and the status of the RDQMs in the HA configuration is displayed, plus their current DR role, for example:

```
Node: main-alice
OS kernel version: 5.14.0-362.18.1
DRBD OS kernel version: 5.14.0-362.18.1
DRBD version: 9.2.7
DRBD kernel module status: Loaded

Queue manager name: RDQM1
Queue manager status: Running elsewhere
HA current location: main-charlie
HA preferred location: main-charlie
HA blocked location: None

Queue manager name: RDQM9
Queue manager status: Running elsewhere
HA current location: main-bob
HA preferred location: main-bob
HA blocked location: None
DR role: Primary

Queue manager name: RDQM7
Queue manager status: Running
HA current location: This node
HA preferred location: This node
HA blocked location: None
DR role: Primary
```

In this example, RDQM7 and RDQM8 are both DR/HA RDQMs, while RDQM1 is an HA RDQM, which is not configured to be able to switch over to a disaster recovery site.

The DRBD kernel module status is one of the following values:

Loaded

Indicates that the DRBD module has been loaded.

Partially loaded

Can occur when the DRBD module has been loaded, but does not function correctly due to a mismatch.

Not loaded

The DRBD module is not loaded. This can be displayed on a newly installed configuration, when no RDQM queue managers have yet been created.

Not installed

Indicates that either the DRBD module is not installed, or that IBM MQ was unable to determine the OS kernel version of the DRBD module.

Previously installed version still loaded

This status can arise if a new DRBD module is installed while the existing DRBD module is running (that is, an RDQM queue manager is running). The newly installed module is reported in the status, but is not the module that is actually running.

- To view the status of a particular queue manager on all the nodes in the HA group, enter the following command:

```
rdqmstatus -m qmname
```

where *qmname* is the name of the RDQM you want to view the status for. The status of the RDQM on the current node is displayed, followed by a summary of the status of the other two nodes from the perspective of the current node.

- To view the status of a particular queue manager on all the nodes in the HA group, including details of any failed resource actions, enter the following command:

```
rdqmstatus -m qmname -a
```

where *qmname* is the name of the RDQM you want to view the status for. The status of the RDQM on the current node is displayed, followed by a summary of the status of the other two nodes from

the perspective of the current node. This is followed by details about any failed resource actions associated with the RDQM.

The following table summarizes the information about the current node that can be returned by the `rdqmstatus -m qmname` command for an RDQM.

Table 34. Current node status		
Status attribute	Possible values	When displayed
Node name	<i>nodename</i>	Always displayed
Queue manager status	state of queue manager (one of the states that are valid for dspmq command)	Always displayed
CPU	<i>n.nn%</i>	Only shown when RDQM is running on this node
Memory	<i>nnn</i> MB used	Only shown when RDQM is running on this node
Queue manager file system	<i>nnn</i> MB used, <i>y.y</i> GB allocated [<i>z%</i>]	Only shown when RDQM is running on this node
HA role	Primary Secondary Unknown	Always displayed
HA status	All nodes in standby This node in standby Remote nodes in standby Mixed	All nodes in standby Current node in standby Both remote nodes in standby Different status for each remote node
 HA TLS encryption	Yes No Not configured	Always displayed. Indicates whether the HA replication connection is configured to be secured with TLS, and if it is, whether the configuration is active between the local node and the remote node.
HA control	Enabled Disabled Unknown	Always displayed. Shows whether RDQM is under Pacemaker control
HA preferred location	None This node Unknown <i>nodename</i>	Always displayed

Table 34. Current node status (continued)

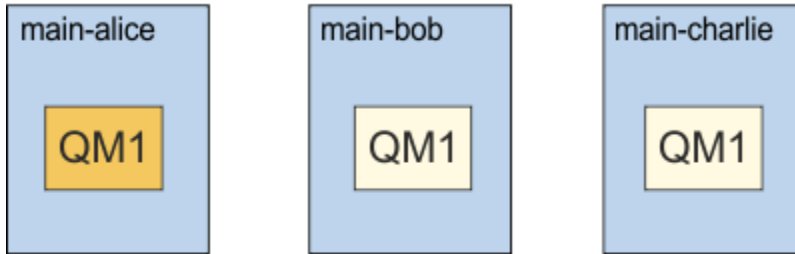
Status attribute	Possible values	When displayed
HA blocked location	None - The queue manager is not blocked from running on any nodes This node - The queue manager is blocked from running on the current node due to one or more failed resource actions <i>nodename</i> - The queue manager is blocked from running on <i>nodename</i> due to one or more failed resource actions <i>nodename1, nodename2</i> - The queue manager is blocked from running on <i>nodename1</i> and <i>nodename2</i> due to one or more failed resource actions All nodes - The queue manager is blocked from running on all nodes due to one or more failed resource actions	Always displayed
HA floating IP interface	<i>Interface_name</i>	Always displayed
HA floating IP address	<i>IPV4_address</i>	Always displayed
DR role	Primary Secondary Secondary pending Unknown	Always displayed
DR status	Normal Synchronization in progress Partitioned Remote system unavailable Inconsistent Reverting to snapshot Remote system not configured Negotiation failed	Everything is OK. Synchronization is in progress. The user has started the queue manager on each node while the DR replication network was unavailable. The connection to the other node has been lost. A synchronization was in progress but was interrupted. The user has chosen to revert to the snapshot that was taken when the queue manager entered the Inconsistent state. The primary has been configured but the secondary has not. The initial negotiation between the primary and secondary nodes failed. This can be caused by incompatible replication types or if the secondary node is configured with a smaller filesystem size.

Table 34. Current node status (continued)		
Status attribute	Possible values	When displayed
DR status (on HA secondary node)	See <i>HA_Primary_Node</i>	Displayed on the HA secondary nodes as the DR status is only known on the HA primary node.
DR port	The TCP/IP port used to replicate the data for this queue manager.	Always displayed.
DR local IP address	The local IP address this queue manager will use for DR replication	Always displayed.
DR remote IP address list	The remote IP addresses this queue manager will use for DR replication. A comma-separated list of three IP addresses.	Always displayed.
DR current remote IP address	The current remote IP this queue manager is connected to for DR replication.	For an HA primary with an active DR connection.
DR current remote IP address (on HA secondary node)	See <i>HA_Primary_Node</i>	Displayed on an HA Secondary node as the DR connection is only on the HA Primary node
 DR TLS encryption	Yes No Not configured See <i>HA_Primary_Node</i>	Always displayed. Indicates whether the DR replication connection is configured to be secured with TLS, and if it is, whether the configuration is active.
DR out of sync data	xKB	Displayed when remote node unavailable or inconsistent.
DR synchronization progress		y%
DR estimated time to completion	YYYY-MM-DD HH:MM:SS	Displayed when a synchronization is in progress.
Snapshot reversion progress	y%	Displayed when DR status is "Reverting to snapshot"
DR last in sync	YYYY-MM-DD HH:MM:SS	Displayed when DR data is out of sync (after initial synchronization). Gives the time and date when the data was last in sync.

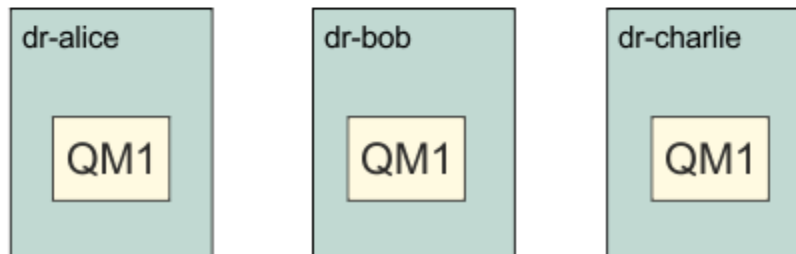
Example

These examples illustrate the command **rdqmstatus -m qm1** run on various nodes of the following DR/HA configuration:

main site



dr site



Example of normal status on a node that is the DR primary and HA primary:

```
Node: main-alice
Queue manager status: Running
CPU: 0.00%
Memory: 123MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
HA role: Primary
HA status: Normal
HA control: Enabled
HA current location: This node
HA preferred location: This node
HA blocked location: None
HA floating IP interface: None
HA floating IP address: None
DR role: Primary
DR status: Normal
DR port: 3000
DR local IP address: 192.168.1.1
DR remote IP address list: 192.168.2.1,192.168.2.2,192.168.2.3
DR current remote IP address: 192.168.2.1
DR TLS encryption: Not configured

Node: main-bob
HA status: Normal
HA TLS encryption: Not configured

Node: main-charlie
HA status: Normal
HA TLS encryption: Not configured
```

Example of normal status on a node that is the DR primary and an HA secondary:

```
Node: main-bob
Queue manager status: Running elsewhere
HA role: Secondary
HA status: Normal
HA control: Enabled
HA current location: main-alice
HA preferred location: main-alice
HA blocked location: None
HA floating IP interface: None
HA floating IP address: None
DR role: Primary
DR status: See main-alice
DR port: 3000
```

```

DR local IP address:      192.168.1.2
DR remote IP address list: 192.168.2.1,192.168.2.2,192.168.2.3
DR current remote IP address: See main-alice v 10.0.0
DR TLS encryption:      See main-alice

Node:                    main-alice
HA status:               Normal
HA TLS encryption:      Not configured

Node:                    main-charlie
HA status:               Normal
HA TLS encryption:      Not configured

```

Example of normal status on a node that is the DR secondary and an HA primary:

```

Node:                    dr-alice
Queue manager status:    Ended immediately
HA role:                 Primary
HA status:               Normal
HA control:              Enabled
HA current location:     This node
HA preferred location:   This node
HA blocked location:     None
HA floating IP interface: None
HA floating IP address:  None
DR role:                 Secondary
DR status:               Normal
DR port:                 3000
DR local IP address:     192.168.2.1
DR remote IP address list: 192.168.1.1,192.168.1.2,192.168.1.3
DR current remote IP address: 192.168.1.1 v 10.0.0
DR TLS encryption:      Not configured

Node:                    dr-bob
HA status:               Normal
HA TLS encryption:      Not configured

Node:                    dr-charlie
HA status:               Normal
HA TLS encryption:      Not configured

```

Example of normal status on a node that is the DR secondary and an HA secondary:

```

Node:                    dr-bob
Queue manager status:    Ended immediately
HA role:                 Secondary
HA status:               Normal
HA control:              Enabled
HA current location:     dr-alice
HA preferred location:   dr-alice
HA blocked location:     None
HA floating IP interface: None
HA floating IP address:  None
DR role:                 Secondary
DR status:               See dr-alice
DR port:                 3000
DR local IP address:     192.168.2.2
DR remote IP address list: 192.168.1.1,192.168.1.2,192.168.1.3
DR current remote IP address: See dr-alice v 10.0.0
DR TLS encryption:      See dr-alice

Node:                    dr-alice
HA status:               Normal
HA TLS encryption:      Not configured

Node:                    dr-charlie
HA status:               Normal
HA TLS encryption:      Not configured

```

Example of DR synchronization in progress on a node that is a DR primary and HA primary:

```

Node:                    main-alice
Queue manager status:    Running
CPU:                     0.00%
Memory:                  123MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
HA role:                 Primary
HA status:               Normal

```

```

HA control: Enabled
HA current location: This node
HA preferred location: This node
HA blocked location: None
HA floating IP interface: None
HA floating IP address: None
DR role: Primary
DR status: Normal
DR port: 3000
DR local IP address: 192.168.1.1
DR remote IP address list: 192.168.2.1,192.168.2.2,192.168.2.3
DR current remote IP address: 192.168.2.1
DR TLS encryption: Not configured
DR synchronization progress: 11.0%
DR estimated time to completion: 2018-09-06 14:55:05

Node: main-bob
HA status: Normal
HA TLS encryption: Not configured

Node: main-charlie
HA status: Normal
HA TLS encryption: Not configured

```

Example of DR partitioned on a node that is a DR primary and HA primary:

```

Node: main-alice
Queue manager status: Running
CPU: 0.00%
Memory: 123MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
HA role: Primary
HA status: Normal
HA control: Enabled
HA current location: This node
HA preferred location: This node
HA blocked location: None
HA floating IP interface: None
HA floating IP address: None
DR role: Primary
DR status: Partitioned
DR port: 3000
DR local IP address: 192.168.1.1
DR remote IP address list: 192.168.2.1,192.168.2.2,192.168.2.3
DR current remote IP address: 192.168.2.1
DR TLS encryption: Not configured
DR out of sync data: 372KB

Node: main-bob
HA status: Normal
HA TLS encryption: Not configured

Node: main-charlie
HA status: Normal
HA TLS encryption: Not configured

```

Example of DR out of sync on a node that is a DR primary and HA primary:

```

Node: main-alice
Queue manager status: Running
CPU: 0.00%
Memory: 123MB
Queue manager file system: 51MB used, 1.0GB allocated [5%]
HA role: Primary
HA status: Normal
HA control: Enabled
HA current location: This node
HA preferred location: This node
HA blocked location: None
HA floating IP interface: None
HA floating IP address: None
DR role: Primary
DR status: Remote unavailable
DR port: 3000
DR local IP address: 192.168.1.1
DR remote IP address list: 192.168.2.1,192.168.2.2,192.168.2.3
DR current remote IP address: Unknown
DR TLS encryption: Not configured
DR out of sync data: 372KB

```

```

DR last in sync:                2020-02-02 20:22:02

Node:                           main-bob
HA status:                      Normal
HA TLS encryption:              Not configured

Node:                           main-charlie
HA status:                      Normal
HA TLS encryption:              Not configured

```

Example of HA out of sync on a node that is a DR primary and HA primary:

```

Node:                           main-alice
Queue manager status:           Running
CPU:                           0.00%
Memory:                         123MB
Queue manager file system:      51MB used, 1.0GB allocated [5%]
HA role:                        Primary
HA status:                      Normal
HA control:                     Enabled
HA current location:            This node
HA preferred location:          This node
HA blocked location:            None
HA floating IP interface:       None
HA floating IP address:         None
DR role:                        Primary
DR status:                      Normal
DR port:                        3000
DR local IP address:            192.168.1.1
DR remote IP address list:      192.168.2.1,192.168.2.2,192.168.2.3
DR current remote IP address:   192.168.2.1
DR TLS encryption:              Not configured

Node:                           main-bob
HA status:                      Inconsistent
HA TLS encryption:              Not configured
HA out of sync data:            15932KB
HA last in sync:                2020-02-02 20:22:02

Node:                           main-charlie
HA status:                      Normal
HA TLS encryption:              Not configured

```

Example of a summary status showing a mismatch between the OS kernel version (RHEL 9.3) and the DRBD kernel module (targeted for RHEL 9.2). Even though the status reports that DRBD kernel module is loaded and queue manager is running, you should update DRBD kernel module with the version targeted for the running OS kernel in this situation.

```

Node:                           main-alice
OS kernel version:              5.14.0-362.18.1
DRBD OS kernel version:         5.14.0-284.11.1
DRBD version:                   9.2.7+ptf.14
DRBD kernel module status:      Loaded

Queue manager name:             QM1
Queue manager status:           Running
HA current location:            This node
HA preferred location:          This node
HA blocked location:            None
DR role:                        Primary

```

Example of a summary status showing a mismatch between the OS kernel version (RHEL 9.3) and the DRBD kernel module (targeted for RHEL 9.0). In this example the version mismatch is more severe and DRBD kernel module fails to load successfully. QM1 is an HA/DR queue manager and moves to another node, its HA status is unknown and its DR status is unknown. To resolve this failure the DRBD kernel module must be updated with the version target for the running OS kernel.

```

Node:                           main-alice
OS kernel version:              5.14.0-362.18.1
DRBD OS kernel version:         5.14.0-70.13.1
DRBD version:                   9.2.7+ptf.14
DRBD kernel module status:      Partially loaded

Queue manager name:             QM1
Queue manager status:           Running elsewhere

```


HA status:	Unknown
HA current location:	main-bob
HA preferred location:	This node
HA blocked location:	None
DR role:	Primary
DR status:	Unknown

Related reference

 [rdqmdrstatus](#)

Operating in a DR/HA environment

When operating in a DR/HA environment there are separate considerations for high availability and disaster recovery.

If the node on which a DR/HA RDQM is running fails, the RDQM automatically fails over to another node in that HA group. If the entire site fails, then you must manually start the RDQM on the preferred node in the HA group on the recovery site. The considerations here are the same as for an ordinary DR RDQM, see [“Operating in a disaster recovery environment”](#) on page 679 for more information.

If one of the nodes fails completely, and needs to be replaced, see [“Replacing a failed node in a disaster recovery configuration”](#) on page 682 and [“Replacing a failed node in a high availability configuration”](#) on page 657 for guidance.

Switching over to a recovery node for planned test or maintenance in a DR/HA configuration

You can perform a manual switch over to test your RDQM disaster recovery and high availability (DR/HA) configuration, or to perform planned maintenance on your main site.

About this task

To test your RDQM DR/HA configuration, you end the primary queue manager at the main site. You then make one of the queue manager instances at the recovery site into the primary and start it. Applications reconnect to the queue manager at the recovery site and the queue manager processes application messages.

Important: You must ensure that the primary instance of a queue manager is stopped and made into a secondary instance, and that it does not fail over to another HA instance, before you promote the secondary instance at the recovery site. Otherwise partitioned data can accrue.

You must either be logged in as root or logged in as a user who belongs to the mqm group and has the necessary sudo configuration.

Procedure

1. Stop the primary instance and re-designate it as the secondary instance:

```
endmqm qmname
rdqmdr -m qmname -s
```

Ending the queue manager in this way stops it failing over to another node in the HA configuration.

2. On the node on your recovery site that has been replicating data from the live site, make the secondary queue manager into the primary by entering the following command on the recovery node:

```
rdqmdr -m qmname -p
```

The queue manager starts.

3. Ensure that your applications reconnect to the queue manager on the recovery queue manager. Provided that you have defined your channels with a list of alternative connection names, specifying your primary and secondary queue managers, then your applications will automatically connect to the new primary queue manager.

What to do next

To return to normal operation, you must stop the queue manager on the recovery node, then promote the queue manager on the original node back to the primary role. Follow the instructions in [Procedure](#), except swapping back from the recovery site to the live site.

Related reference

[strmqm](#)

[rdqmdr](#)

Related information

[endmqm](#)

Switching over to a recovery node if a disaster occurs in a DR/HA configuration

If a disaster occurs in your main site, you take steps to switch over to your recovery site.

About this task

Following the loss of the primary queue manager at the main site, you make the secondary queue manager at the recovery site into the primary and start it. Applications reconnect to the queue manager at the recovery site and the queue manager processes application messages.

You should try to ensure that the primary queue manager at the main site is either unable to run, or has been stopped before you promote the original secondary instance. You should also check that it has not failed over to another HA node at the live site. Otherwise partitioned data can accrue. Similarly, when the main site recovers, if you can access the site, you should try to ensure that the queue manager does not restart and run as a primary instance on any of the HA nodes. If this is not possible, block the replication traffic between the two sites until you can restore the correct primary/secondary relationship.

You must either be logged in as root or logged in as a user who belongs to the mqm group and has the necessary sudo configuration.

Procedure

1. On the node on your recovery site that has been replicating data from the live site, make the secondary queue manager into the primary by entering the following command on the recovery node:

```
rdqmdr -m qmname -p
```

The queue manager starts.

2. Ensure that your applications reconnect to the queue manager on the recovery queue manager. Provided that you have defined your channels with a list of alternative connection names, specifying your primary and secondary queue managers, then your applications will automatically connect to the new primary queue manager.

What to do next

When the failed site is restored, if you can access it, try to ensure that the queue manager does not restart and resume running on any of the nodes in the HA configuration. If any partitioned data has accrued, take steps to resolve this situation, see [“Resolving a partitioned \(split brain\) problem in DR RDQM” on page 684](#). To return to normal operation, you must stop the queue manager on the recovery node, then promote the queue manager on one of the nodes at the original live site back to the primary role. Follow the instructions in [Procedure](#), except swapping back from the recovery site to the live site.

Related reference

[strmqm](#)

[rdqmdr](#)

Replacing a failed node in a DR/HA configuration

If one of the nodes in either of your HA groups fails, you can replace it.

About this task

The procedure varies according to whether the node that you are replacing is a primary or a secondary in the DR configuration. In either case, the new node must have an identical configuration to the node that you are replacing, that is, it must have the same hostname, same IP addresses, and so on.

You might also encounter the situation where you have completely lost the HA group at your main or recovery site and have to replace the entire HA group.

Procedure

- For a replacement node that is a primary in the DR configuration, complete the following steps on the new node:
 - a) Create an `rdqm.ini` file that matches the files on the other nodes, and then run the `rdqmadm -c` command (see [“Defining the Pacemaker cluster \(HA group\)”](#) on page 630).
 - b) Run the `crtmqm -sxs -rr p qmanager` command to recreate each DR/HA RDQM (see [“Creating DR/HA RDQMs”](#) on page 692).
- For a replacement node that is a secondary in the DR configuration, complete the following steps on the new node:
 - a) Create an `rdqm.ini` file that matches the files on the other nodes, and then run the `rdqmadm -c` command (see [“Defining the Pacemaker cluster \(HA group\)”](#) on page 630).
 - b) Run the `crtmqm -sx -rr s qmanager` command to recreate each DR/HA RDQM (see [“Creating DR/HA RDQMs”](#) on page 692).
- To replace an entire HA group, complete the following steps:
 - a) If you lose the entire HA group at the DR primary site (that is, the main site), then you must follow the steps to perform a managed failover to the DR secondary site to keep running your DR/HA RDQMs (see [“Switching over to a recovery node if a disaster occurs in a DR/HA configuration”](#) on page 710).
 - b) Recreate the HA group on your three replacement nodes, as described in [“Configuring HA groups for DR/HA RDQMs”](#) on page 687.
 - c) Recreate your DR/HA RDQMs on the new HA group as described in [“Creating DR/HA RDQMs”](#) on page 692.
 - d) If required, perform a managed failover from your recovery site back to your main site.

DR/HA RDQM worked example

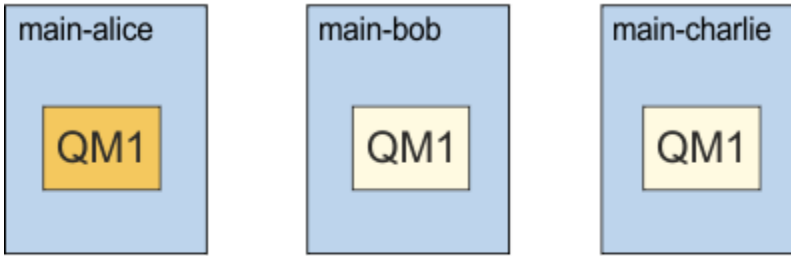
This example show how to create and delete DR/HA RDQM.

Creating a DR/HA RDQM

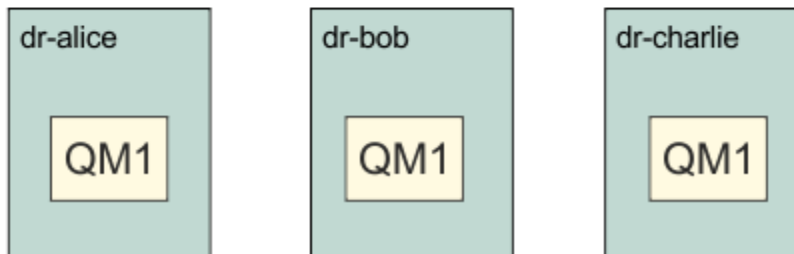
The example configuration has two sites, named 'main' and 'dr'. Each site has three nodes, named 'alice', 'bob', and 'charlie'. The nodes have full name consisting of their site name and name, so 'main-alice', 'dr-alice', and so on.

The following steps create a DR/HA RDQM named QM1 that runs on main-alice. The main-alice node is the HA and DR primary.

main site



dr site



If the local and remote DR IP addresses are specified in the `rdqm.ini` file, then there is not need to specify any IP addresses on the command line and a DR/HA RDQM named QM1 can be created by running the following command on main-alice:

```
crtmqm -sx -rr p -rn DR1 -rp 7001 QM1
```

If the local DR IP addresses are specified in the `rdqm.ini` file, then the remote DR IP addresses can be specified on the command line:

```
crtmqm -sx -rr p -ri 192.168.2.1,192.168.2.2,192.168.2.3 -rp 7001 QM1
```

If no DR IP addresses are specified in the `rdqm.ini` file, then both remote and local DR IP addresses can be specified on the command line:

```
crtmqm -sx -rr p -rl 192.168.1.1,192.168.1.2,192.168.1.3 -ri  
192.168.2.1,192.168.2.2,192.168.2.3 -rp 7001 QM1
```

The output in response to creating QM1 is as shown in the following example:

```
Creating replicated data queue manager configuration.  
Secondary queue manager created on 'main-bob'.  
Secondary queue manager created on 'main-charlie'.  
IBM MQ queue manager created.  
Directory '/var/mqm/vols/qm1/qmgr/qm1' created.  
The queue manager is associated with installation 'Installation1'.  
Creating or replacing default objects for queue manager 'QM1'.  
Default objects statistics : 83 created. 0 replaced. 0 failed.  
Completing setup.  
Setup completed.  
Enabling replicated data queue manager.  
Replicated data queue manager enabled.  
Issue the following command on the remote HA group to create the DR/HA secondary queue manager:  
crtmqm -sx -rr s -rl 192.168.2.1,192.168.2.2,192.168.2.3 -ri  
192.168.1.1,192.168.1.2,192.168.1.3 -rp 7001 -fs 3072M QM1
```

Copy the command from the message to create the DR secondary instance of QM1 on dr-alice:

```
crtmqm -sx -rr s -rl 192.168.2.1,192.168.2.2,192.168.2.3 -ri  
192.168.1.1,192.168.1.2,192.168.1.3 -rp 7001 -fs 3072M QM1
```

The following message is output on dr-alice:

```
Creating replicated data queue manager configuration.  
Secondary queue manager created on 'dr-bob'.  
Secondary queue manager created on 'dr-charlie'.  
IBM MQ secondary queue manager created.  
Enabling replicated data queue manager.
```

Test the DR Secondary

To test the disaster recovery features of QM1, run the following command on main-alice to make QM1 the DR secondary instance:

```
rdqmdr -m QM1 -s  
Queue manager 'QM1' has been made the DR secondary on this node.
```

Run the following command on dr-alice to make QM1 the DR primary instance on that node:

```
rdqmdr -m QM1 -p  
Queue manager 'QM1' has been made the DR primary on this node.
```

Deleting a DR/HA RDQM

To delete the DR/HA RDQM named QM1, first end the queue manager on main-alice:

```
endmqm -w QM1  
Replicated data queue manager disabled.  
Waiting for queue manager 'QM1' to end.  
IBM MQ queue manager 'QM1' ended.
```

Then run the following command on main-alice to delete QM1:

```
dltmqm QM1  
Removing replicated data queue manager configuration.  
Secondary queue manager deleted on 'main-bob'.  
Secondary queue manager deleted on 'main-charlie'.  
IBM MQ queue manager 'QM1' deleted.
```

Finally, you must delete QM1 on dr-alice:

```
dltmqm QM1  
Removing replicated data queue manager configuration.  
Secondary queue manager deleted on 'dr-bob'.  
Secondary queue manager deleted on 'dr-charlie'.  
IBM MQ queue manager 'QM1' deleted.
```

Related concepts

[“Operating in a DR/HA environment” on page 709](#)

When operating in a DR/HA environment there are separate considerations for high availability and disaster recovery.

Related tasks

[“Creating DR/HA RDQMs” on page 692](#)

You use the **crtmqm** command to create a replicated data queue manager (RDQM) in a DR/HA configuration.

[“Deleting a DR/HA RDQM” on page 698](#)

You use the **dlrmqm** command to delete a DR/HA replicated data queue manager (RDQM).

Combining highly available queue managers with IBM MQ clusters

High availability mechanisms such as Native HA queue managers and multi-instance queue managers, can be combined with IBM MQ clusters to maximize both service and message availability.

Service availability vs message availability

Service availability refers to whether applications have access to a queue manager with the correct queue definition. This is different from message availability, where applications need access to specific messages on those queues.

- IBM MQ clusters are particularly useful for enabling high service availability since they ensure that applications can always find the queue they need on at least one queue manager, even if the one they were previously talking to is unavailable.
- Highly available queue managers using mechanisms such as Native HA, or multi-instance, ensure that the specific messages on a given queue manager are made accessible as soon as possible following a failure. As such, they are focused on message availability.

By combining both IBM MQ clusters and highly available queue managers, a solution can optimize both service and message availability.

IBM MQ clusters increase service availability

You can increase service availability by creating definitions of the same queue on multiple queue managers within a cluster. To ensure these queues are all known to the cluster, they should be created as “cluster queues”.

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 the same queue definition.

Combining IBM MQ clusters and high availability queue managers

Although clusters greatly increase service availability, there are two related failure scenarios that result in messages getting delayed and therefore reduced message availability.

Marooned messages

If a queue manager in the cluster fails, messages that have already been sent are marooned until the failed queue manager becomes available again.

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? Do you need to receive replies from the same queue manager that the original message was sent to?

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 you have to wait for replies on a specific queue manager, your application is not able to progress until that queue manager is available again.

Building a cluster with highly available queue managers reduces the time a message is delayed, thereby improving message availability.

If the cluster is configured with highly available queue managers, the delay waiting for the failed queue manager to restart is reduced to the time it takes for the standby to take over. This is considerably faster than the time it takes to get the original queue manager instance back into action, and could be as low as a few seconds.

The active and passive members of a high availability queue manager can be spread across the servers used in a cluster such that, should a server fail, all the active instances of queue managers on that server will immediately failover to another server.

Benefits of IBM MQ clusters in relation to availability

As well as having the benefits of reduced administration and increased scalability, clusters provide additional elements of availability to complement highly available queue managers. Clusters enable the following features:

Uninterrupted service

Highly available queue managers on their own do not provide a completely uninterrupted service because it takes time for failover to the standby queue manager to occur, for its channels to be reconnected, and for failed batches of messages to be resubmitted. A cluster provides an uninterrupted service. New messages received by the cluster are sent to active queue managers to be processed.

Localized outage

There are practical limitations to how far apart the active, standby, and file system servers can be, as they need to synchronously replicate between one another.

Clustered queue managers are not replicas of one another, so they 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.

Configuring recovery and restart

You can make your applications resilient by recovering messages after server or storage failure.

About this task

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 **dspmqste** 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 maintains 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.



CAUTION: You cannot move recovery logs to a different operating system.

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 35 on page 716 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 36 on page 717 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 replicated data queue manager, or an HA queue manager on an IBM MQ Appliance, you can specify a floating IP address used by both active and standby queue managers to simplify configuration.
- 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 35. 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

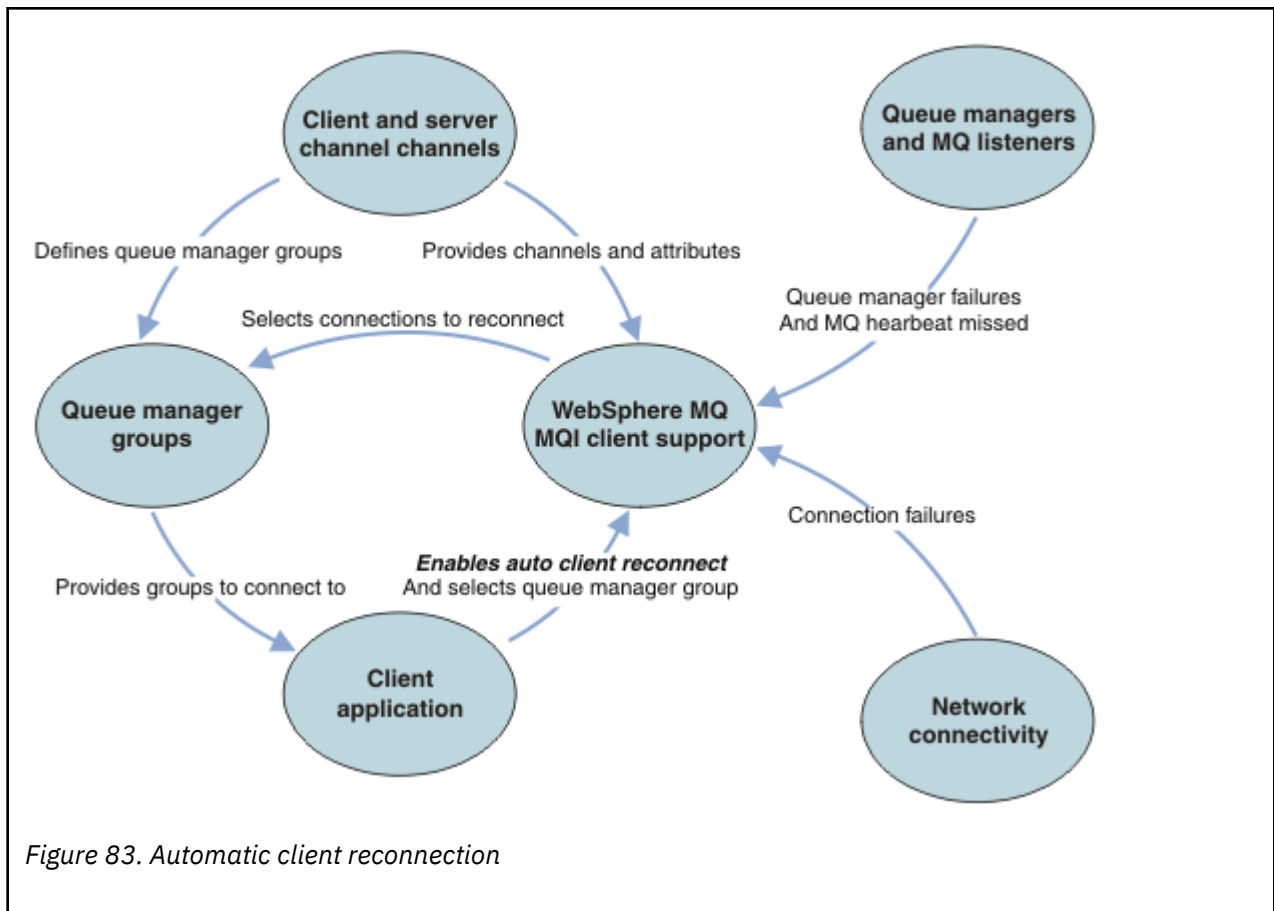
Table 35. Supported clients (continued)			
Client interface	Client	Program access to reconnection options	Reconnection support
Other APIs	Windows Communication Foundation (Unmanaged ¹)	Not supported	7.0.1
	Windows Communication Foundation (Managed ¹)	Not supported	Not supported
	Axis 1	Not supported	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 36. Automatic reconnection configuration requirements		
Component	Requirement	Effect of not meeting requirement
IBM MQ MQI client installation	See Table 35 on page 716	MQRC_OPTIONS_ERROR
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: - MQCONN 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 83 on [page 718](#) shows the main interactions between components that are involved in client reconnection.



Client application

The client application is an IBM MQ MQI client. For details about automatic client reconnection for a JMS client, see [Using automatic JMS client reconnection](#).

- By default clients are not automatically reconnected. Enable the automatic client reconnection by setting the MQCONN 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. Asynchronous consumers should check whether they have been 'suspended' if a transaction is backed out.
 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 MQCONN 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 595](#).

- 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 MQCONN call itself is not tried again if it fails. For example, if you receive the return code 2543 - MQRC_STANDBY_Q_MGR from MQCONN, 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.

High availability queue managers

High availability queue managers have an active instance and one or more standby instances of a queue manager. The active queue manager is synchronized with the standby queue managers, such that a standby can automatically take over if the active instance fails. There are a number of different solutions to provide high availability queue managers, see [“Configuring high availability and disaster recovery” on page 477](#).

You can simplify restarting IBM MQ MQI client applications, after a high availability queue manager has activated its standby instance, by using automatic client reconnection.

The standby instance of a high availability 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. Replicated data queue managers and IBM MQ Appliance high availability queue managers support floating IP addresses, where you specify a single address for use with active or standby queue managers.

Queue manager groups

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 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](#).

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 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 KAINTE 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 MQCONNEX 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 MQCONNEX 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 597.
- 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.

Related concepts

[Reconnectable clients](#)

Related tasks

[Stopping a queue manager](#)

z/OS

Console message monitoring

On IBM MQ for z/OS, 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 can 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.

Console messages to monitor

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.

CSQE222E: Dynamic allocation of SMDS(*qmgr-name*) CFSTRUCT(*struc-name*) data set *dsname* failed with return code *ret-code*, reason code *eeeeiii*

- The queue manager attempted to allocate the shared message data set using the data set name formed by inserting the queue manager name into the generic **DSGROUP**, but the DYNALLOC macro returned an error.
- Check the job log for dynamic allocation error messages giving more details about the problem.
- After any changes, use the **START SMDSCONN** command to trigger a new attempt to use the data set.

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.

CSQM062I: csect-name INDXTYPE(index-type) not allowed for shared transmission queue shared-xmitq

- Change the **INDXTYPE** to either NONE or MSGID.

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 data set. 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 data set 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 is defined on a non-recoverable CF structure

- Follow the system programmer response.

CSQX130E: 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 system 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.

CSQX548E: csect-name Messages sent to local dead-letter queue, channel channel-name reason=mqrc (mqrc-text)

- Follow the system programmer response.

CSQX788I: csect-name DNS lookup for address address using function 'func' took n seconds

- Follow the system programmer response.

CSQY222E: csect-name Queue manager is critically short of local storage - take action

- The queue manager is running critically short of virtual storage in the extended private region. Action should be taken to relieve the situation, and to avoid the possible abnormal termination of the queue manager.

CSQY225E: csect-name Queue manager is critically short of local storage above the bar - take action

- The queue manager is running critically short of virtual storage above the bar. Action should be taken to relieve the situation, and to avoid the possible abnormal termination of the queue manager.

CSQ5038I: csect-name Service task service-task has been unresponsive since hh.mm.ss.nnnnnn. Check for problems with Db2

- Follow the system programmer response.

Logging: Making sure that messages are not lost

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.



CAUTION: You cannot move recovery logs to a different operating system.

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 [“LogDefaults stanza of the mqs.ini file”](#) on page 103), 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 727](#) for more information.

ALW In IBM MQ for AIX or Linux systems, if you have not changed the log path, log extents are created under the directory:

```
/var/mqm/log/QMgrName
```

Windows 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
```

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 configuration information in .ini files on Multiplatforms” on page 94](#).

Log extents are prefixed with either the letter S or the letter R. Active, inactive, and superfluous extents are prefixed with S, whereas reuse extents are prefixed with R.

When backing up or restoring your queue manager, back up and restore all the active, inactive, and superfluous extents, along with the log control file.

Note: You do not need to back up and restore reuse extents.

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 `amqh1ctl1.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. A third type of logging, replicated, is used only by Native HA configurations.

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 [“Using checkpointing to ensure complete recovery” on page 730](#).

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.

You can change an existing log type for a queue manager, from linear to circular, or from circular to linear using the `migmqlog` command.

Replicated logging



Use replicated logging to configure a Native HA configuration. When you create a Native HA group you create three queue managers on different nodes. You specify a logging type of replicated together with a unique instance name for each of the queue managers. The Native HA configuration provides a high availability solution by having an active instance replicate log data to two replica instances. If the active instance fails, one of the replica instances takes over the active role. Log replication ensures that little, if any, data is lost. See [“Native HA” on page 480](#) for more details. A replicated log is equivalent to a linear log with automatic log management and automatic media images enabled.

Linear log extents that are not active



If you are using automatic log management, including archiving, the logger keeps track of linear log extents that are not active.



Attention: If you are using automatic log management, without archiving, the use of a backup queue manager is not supported for this process.



When a log extent is no longer required for recovery and, if necessary, is archived, the logger will, at a convenient point, either delete the log extent or reuse it.

A reused log extent is renamed to be the next in the log sequence. Message AMQ7490 is periodically written, indicating how many extents have been created, deleted, or reused.

The logger chooses how many extents to keep ready for reuse and when to delete those extents.

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 733 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 739 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: All types of logging can cope with unexpected loss of power, assuming that there is no hardware failure.

Moving from circular to linear logs

You can change the type of log your configuration uses by using the **migmqllog** command.

Linear logs enable the restoration of lost or damaged data objects by using the **rcrmqobj** command. However, when converting from circular to linear logging, you should consider the effect that the change has on disk space requirements and performance. In particular, you should consider the following points:

- Linear logs have to accommodate media images, which are not supported in circular logs, and so linear logs are larger.
- The quantity of data written to linear logs depends on the following factors:
 - How often media images are recorded.
 - How much data there is in the queue files at the times media images are written for those queues (deep queues are not good practice, but sometimes cannot be avoided). More data in the queue files means more data must be written out to the log in media images.
 - The normal persistent messages being put/got by the applications.
- Because of the size requirements of linear logs, give attention to the following points:
 - Size the disk partition appropriately. The requirement can be significantly greater than with circular logs (depending on how much data is in the queue files at the times when their media images are being written to the logs).
 - Be aware that normal transaction/persistent messaging performance will vary, and is lower when media images are being recorded.
- Because the normal application transaction/persistent messaging work is now contending with more data being written into the log from media images, then, if the amount of data written to the log by media imaging is high, the following points might apply:

- The applications might experience more rollbacks than with the equivalent circular log.
- The general performance of persistent messaging is lower.

Related information

[migmqlog \(migrate IBM MQ logs\)](#)

Using checkpointing to ensure complete 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 logger schedules checkpoints more frequently (so the next one is scheduled before the previous one has completed) because the logger is attempting to keep the active log in the primary log extents. If this is not possible, an [AMQ7466](#) error is logged.

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 at the following times:

- When the queue manager starts
- At shutdown
- When logging space is running low
- **Multi** After 50,000 operations have been logged since the previous checkpoint was taken
- **z/OS** After *number_of_operations* have been logged since the previous checkpoint was taken, where *number_of_operations* is the number of operation set in the **LOGLOAD** property.

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.

Multi 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.

Multi 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.

Multi 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=000000001E45FF7AD to RBA=000000001E48C1882
CSQR005I +MQOX RESTART - FORWARD RECOVERY COMPLETE - 816
IN COMMIT=0, INDOUBT=0
CSQR032I +MQOX Backward recovery log range 817
from RBA=000000001E48C1882 to RBA=000000001E48C1882
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 84 on page 732, 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.

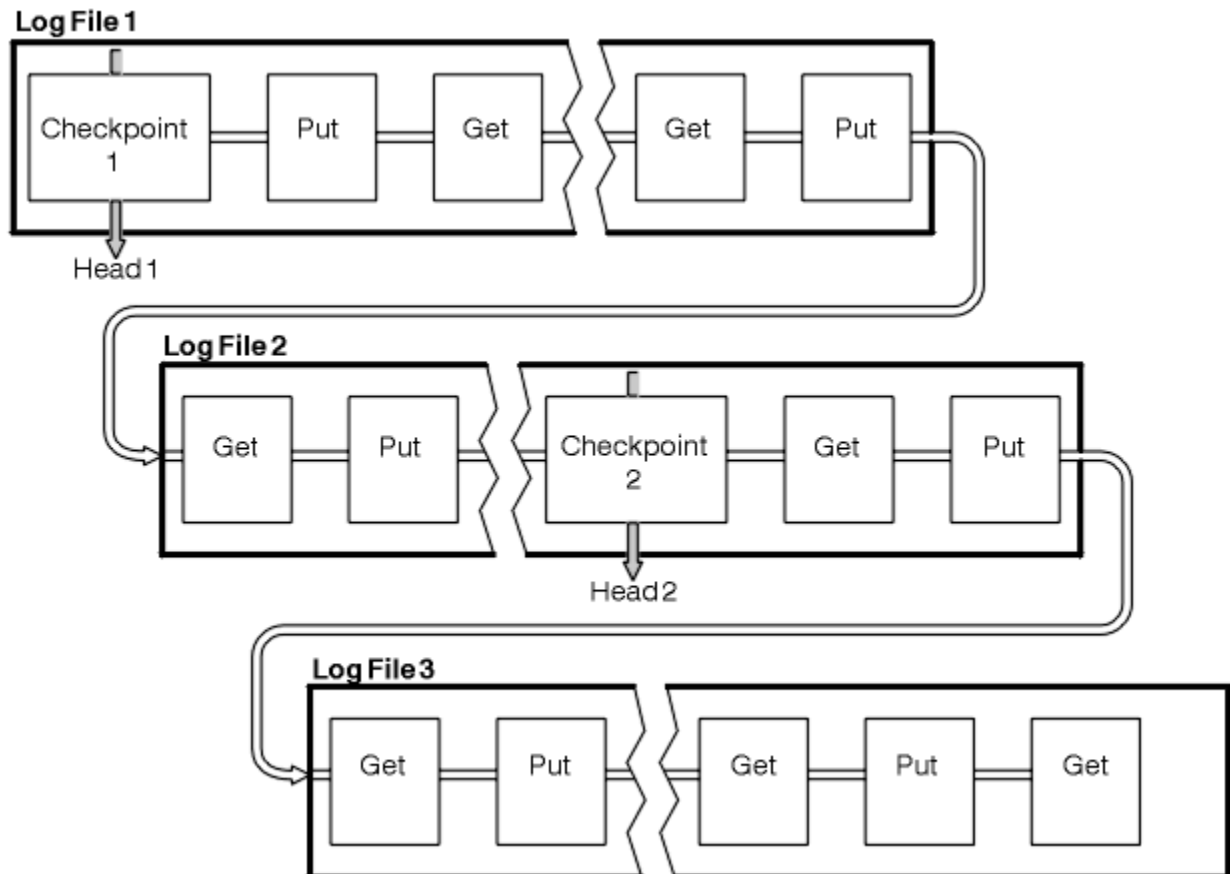


Figure 84. Checkpointing

Checkpointing with long-running transactions

How a long-running transaction affects reuse of log files.

Figure 85 on page 733 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.

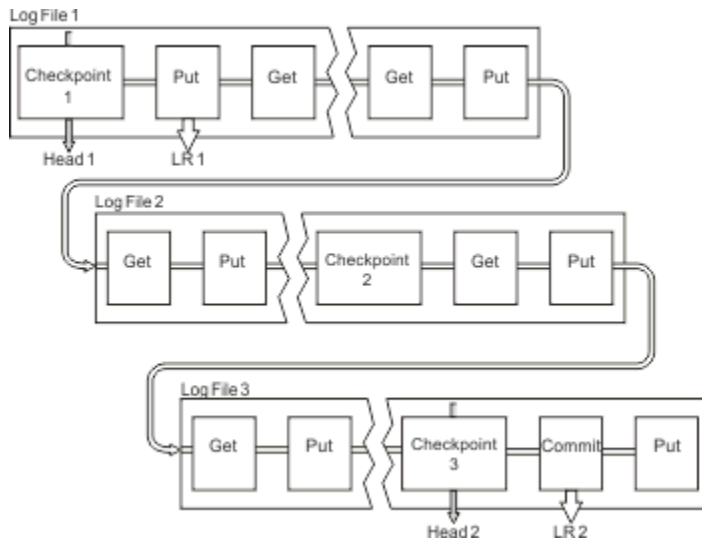


Figure 85. Checkpointing with a long-running transaction

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 AIX 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 734 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 into consideration before taking a checkpoint because the amount of log space is running low.

The queue manager attempts to schedule checkpoints in a manner that keeps the log usage within the primary extents.

See “LogDefaults stanza of the mq.ini file” on page 103 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 **dspmqttrn -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.

You can discover the bytes per second that the queue manager is writing to the log by issuing the following command:

```
amqsrua -m qmgr -c DISK -t Log
```

The logical bytes written, shows the bytes per second that the queue manager is writing to the log. For example:

```
$ amqsrua -m mark -c DISK -t Log
Publication received PutDate:20160920 PutTime:15383157 Interval:4 minutes,39.579 seconds
Log - bytes in use 37748736
Log - bytes max 50331648
Log file system - bytes in use 316243968
Log file system - bytes max 5368709120
Log - physical bytes written 4334030848 15501948/sec
Log - logical bytes written 3567624710 12760669/sec
Log - write latency 411 uSec
```

In this example, the logical bytes per second written to the log is 12760669/sec or approx 12 MiB per second.

Using

```
DISPLAY CONN(*) UOWLOGDA UOWLOGTI
```

showed 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 MiB
```

.

The number of log files must be able to contain the largest expected log size (calculated in the preceding text). This will be:

Minimum number of log files = (Required log size) / (**LogFilePages** * log file page size (4096))

Using the default **LogFilePages**, which is 4096, and the log size estimate of 264MiB, calculated in the preceding text, the minimum number of log files should be:

```
264MiB / (4096 x 4096) = 16.5
```

that is, 17 log files.

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 so the workload runs within the primary log files:

- **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.

The logger attempts to keep the workload in the primary files alone. Therefore, the logger schedules checkpoints when a fraction of the primary files alone are full.

Having the secondary files is a contingency, in case there are any unexpectedly long running transactions.

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.
- The `LOGUTIL` field of `DISPLAY QMSTATUS LOG` is available to help you calculate, approximately, the size of active log required.

This field is designed to allow you to make a reasonable estimate of the required log size without constantly sampling in order to determine the duration of your longest running transactions, or the peak throughput of the queue manager.

How large should I make my LogFilePages?

Generally make your LogFilePages large enough so that you are able to easily increase the size of your active log without reaching the maximum number of primary files. A few large log files is preferable to many small log files because a few large log files allows you more flexibility to increase the size of your log should you need to.

For linear logging, very large log files might make the performance variable. With very large log files there is a bigger step to create and format a new log file, or to archive an old one. This is more of a problem with manual and archive log management because with automatic log management new log files are rarely created.

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.
- 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 [“Using checkpointing to ensure complete recovery”](#) on page 730 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 transactions are tolerated.

Important: No matter how inaccurately you estimate the size of your log, data integrity is maintained.

To help you estimate the maximum size of the log, you can use the log utilization statistics. For further information, see [“Deciding how to set IMGLOGLN and IMGINTVL” on page 744](#) and [ALTER QMGR](#).

See [“Using checkpointing to ensure complete recovery” on page 730](#) 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.

How large should I make my log filesystem?

Estimating the size of log filesystem a queue manager needs.

It is important that you make your log filesystem large enough, so that your queue manager has plenty of space to write its log. If the queue manager fills the log filesystem completely, it will write FFDCs, rollback transactions, and might terminate the queue manager abruptly.

The amount of disk space you reserve for your log must be at least as large as the active log. Exactly how much larger depends on:

- Your choice of log type (linear or circular)
- The size of the active log (primary files, secondary files, log file pages)
- Your choice of log management (manual, automatic, or archive)
- Your contingency plans in the case of a damaged object.

If you choose a circular log then your log filesystem should be

```
LogFileSystemSize >= (PrimaryFiles + SecondaryFiles + 1) * LogFileSize
```

where $\text{LogFileSize} = \text{logFilePages} * 4096$.

This allows the queue manager to write to all the primary and secondary files. In exceptional circumstances, the queue manager might write an extra extent beyond the number of secondaries. The preceding algorithm takes that into account.

If you choose a linear log, the log filesystem should be significantly larger than the active log.

If you choose manual log management, the queue manager continues to write to new log extents as it needs them, and it is your responsibility to delete them (and archive them) when they are no longer required.

How much larger the log filesystem needs to be depends largely on your strategy for deleting superfluous or inactive extents.

You might decide to archive and delete extents as soon as they become inactive (not needed for restart recovery) or you might decide to archive and delete only superfluous extents (not needed for media or restart recovery).

If you are archiving and deleting only superfluous extents, and if you have a damaged object, **MEDIALOG** will not move forward, so no more extents will become superfluous. You will stop archiving and deleting extents until you resolve the problem, perhaps by recovering the object.

Unless you stop the workload, how much time you have to resolve the problem depends on the size of your log filesystem. Hence, it is best practice to have a generous log filesystem when using linear logging.

If you choose a linear log and automatic or archive log management, the queue manager reuses log extents.

Log extents that are available to be reused are prefixed with the letter R. When a media image is recorded, as superfluous extents are archived, the queue manager can then reuse those extents.

So the reuse extents are less than the data length written to the log between media images:

```
ReuseExtents <= LogDataLengthBetweenMediaImages
```

When recording media images automatically and setting **IMGLOGLN**, `LogDataLengthBetweenMediaImages` can be as much as twice **IMGLOGLN** because **IMGLOGLN** is a target not a fixed maximum.

When manually recording media images, or recording them automatically by interval, `LogDataLengthBetweenMediaImages` depends on your workload and the interval between taking images.

In addition to active extents and reuse extents, there are inactive extents (needed for media recovery only) and superfluous extents (not needed for restart or media recovery).

When using automatic or archive log management, the queue manager does not reuse extents that are needed for media recovery. So, the number of inactive extents depends on how frequently you are taking media images, and whether you are taking them manually or automatically.

IMGINTVL and **IMGLOGLN** are targets, not a fixed minimum or maximum between media images.

However when estimating the maximum size of log filesystem you might need, it is unlikely that automatic media images would be recorded more than twice **IMGINTVL** or **IMGLOGLN** apart.

When sizing your log filesystem using automatic or archive log management, you should also consider what might happen if a queue or other object is damaged. In that case, the queue manager is not able to take a media image of the damaged object, and **MEDIALOG** will not move forward.

If your workload continues, your inactive log will grow unrestrained as the oldest extent needed for media recovery is still needed and cannot be reused. If your workload continues, you will have until your log filesystem fills completely to fix the problem, before the queue manager starts rolling back transactions and might even end abruptly.

Hence for automatic and archive log management:

```
LogFilesystemSize > (PrimaryFiles + SecondaryFiles +
(((TimeBetweenMediaImages *2) + TimeNeededToResolveDamagedObject) * ExtentsUsedPerHour))
* LogFilePages
```

Note: The preceding algorithm assumes that **SET LOG ARCHIVED** is called for each extent, as soon as it is no longer needed for media recovery, for archive log management.

Managing logs

The product supports automatic log management and automatic media recovery of linear logs. Circular logs are nearly self-managing, but sometimes need intervention to resolve space problems.

Note:  Automatic and archive log management are not valid on IBM i.

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.

You can delete secondary files when using circular logging. See [RESET QMGR TYPE \(REDUCELOG\)](#) for more information.

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.

 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.

An application that has a transaction rolled back in this way cannot perform subsequent **MQPUT** or **MQGET** operations specifying sync point under the same transaction.

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.

Automatic media imaging

If you are using linear logging, you can periodically take media images. These images allow you to recover objects if they are damaged.

By default, scheduling media imaging is a manual process. You can also configure media imaging to be automatic. If you configure automatic media imaging, the queue manager schedules the imaging when workload permits. Administration is simplified and performance impact is minimized.

You enable automatic media imaging, and define how a queue manager implements it, by setting the following queue manager attributes by using the **ALTER QMGR MQSC** command:

IMGSCHED

Set this attribute to AUTO to enable automatic media imaging for the queue manager. If you enable automatic media imaging, use these two attributes to tune when images are taken:

IMGINTVL

Specifies a target frequency for recording media images. Specify a value in minutes, the default is 60 minutes. Set to OFF to not use time interval to influence media imaging.

IMGLOGLN

Specifies the target amount of log written after which media images are taken. Specify a value in number of megabytes, Set to OFF to not use amount of log written to influence media imaging. The default is OFF.

IMGRCOVO

Use this attribute to specify which types of object are recoverable. Set this attribute to YES to take media images of the authentication information, channel, client connection, listener, namelist, process, alias queue, remote queue, and service objects.

IMGRCOVQ

Use this attribute to set the default for local and PERMADYN queue types. Set to YES to take media images, NO otherwise.

Related information

[ALTER QMGR \(alter queue manager settings\)](#)

Automatic log management

You can configure automatic log management for a linear log so that log extents can be reused.

By default, a linear log must be managed manually, which means you must delete or archive log files that are no longer required for recovery in order to avoid running out of disk space.

If you enable automatic log management then log extents are automatically recycled as soon as possible. Automatic log management reduces the possibility of running out of space but you still must provide sufficient spare disk capacity. You must provide enough headroom in case media images consume a lot of disk space. The amount of spare disk space needed depends on the size of media images, which can vary. See [“Managing log files” on page 742](#) for guidance.

You can specify the logging management settings when you create a queue manager, or you can edit the `qm.ini` file to change the logging management for an existing queue manager. You can select one of three log management modes:

Manual

Log extents are managed manually. This is the default setting.

Automatic

The queue manager automatically reuses log extents when they are no longer required for restart or media recovery

Archive

This is the same as Automatic except that the queue manager waits for notification that an extent has been archived before reusing it.

You specify logging management settings when creating a queue manager by using the following arguments with the **crtmqm** command:

-lla

Specifies that automatic log management is used.

-lln

Specifies that archive log management is used.

-ll

Specifies that manual log management is used (the default).

For an existing queue manager, you can change the log management settings by editing the Log stanza of a queue manager's `qm.ini` file. Add one of the following settings:

LogManagement=Automatic

Specifies that automatic log management is used.

LogManagement=Archive

Specifies that archive log management is used.

If you select the archive log management method, you must inform the queue manager when a log extent is archived. You do this by using the following command:

```
SET LOG ARCHIVED(logfilename)
```

You can also use the PCF command **MQCMD_SET_LOG**.

Related information

[crtmqm \(create queue manager\)](#)

[SET LOG \(notify completion of log archiving\) on Multiplatforms](#)

[MQCMD_SET_LOG \(notify completion of log archiving\) on AIX, Linux, and Windows](#)

What happens when a disk gets full

When a queue manager is configured to use linear logging, the queue manager logging component reacts to a disk full condition in the following ways.

If the disk containing the log files becomes full, then:

- The queue manager discovers this condition only when creating a new log file of the required size, which it does in advance of when it is needed.
- It discovers the disk full condition when the operating system returns an error from the request to extend the file to the required size.
- The queue manager issues message AMQ6708 to the queue manager error log.
- A First Failure Support Technology (FFST) record is written into the system-wide errors directory. This record provides details of the disk full condition and should be retained should you need to contact IBM Support.

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.

When you use linear logging, you have the option to use:

- Automatic management of log extents.

See [DISPLAY QMSTATUS](#) for more information on the new log attributes.

Also, see the following commands, or their PCF equivalents:

- [RESET QMGR](#)
- [SET LOG](#) for distributed platforms

- The options controlling the use of media images.

See the [ALTER QMGR](#) command and [ALTER QUEUES](#) for more information on:

- [IMGINTVL](#)
- [IMGLOGLN](#)
- [IMGRCOVO](#)
- [IMGRCOVQ](#)
- [IMGSCHED](#)

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 [“LogDefaults stanza of the mqs.ini file” on page 103](#) and [“Log stanza of the qm.ini file” on page 142](#)). 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 methods determine 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 [Administering IBM MQ using 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.

Cleaning log extents automatically - linear logging only



You have the option to use automatic management of linear log extents no longer required for recovery.

You use the **LogManagement** attribute in the Log stanza of the qm.ini file, or by using the IBM MQ Explorer, to set up automatic management. See [“Log stanza of the qm.ini file” on page 142](#) for more information.

See the LOG parameter of **DISPLAY QMSTATUS** for more details about the operation of the log, and the following commands for using the log:

- [RESET QMGR](#)
- [SET LOG](#)

Taking media images automatically - linear logging only

There is an overall switch to control whether the queue manager automatically writes media images, the default being that the switch has not been set.

You can control whether automatic media imaging occurs, and the frequency of the process, by using the following queue manager attributes:

IMGSCHED

Whether the queue manager writes media images automatically

IMGINTVL

Frequency for writing media images, in minutes

IMGLOGLN

Megabytes of log written since previous media image of an object.

If you have a critical time during the day when workload is very heavy and you want to be sure that system throughput is not impacted by taking automatic media images, you might wish to temporarily switch off automatic media imaging by setting **IMGSCHED(MANUAL)**.

You can switch **IMGSCHED** at any time during the workload.



Attention: MEDIALOG is not moved forward if you are not taking media images, so you must either archive extents, or ensure you have sufficient disk space.

You can also control automatic and manual media images for other user-defined objects by using the **IMGRCOVO** attribute:

- Authentication information
- Channel
- Client connection
- Listener
- Namelist
- Process
- Alias queue
- Local queue
- Service
- Topic

For internal system objects, such as the object catalog and the queue manager object, the queue manager automatically writes media images as appropriate.

See [ALTER QMGR](#) for more information on the attributes.

You can also enable or disable automatic and manual media images for local and permanent dynamic queues only. You do this using the **IMGRCOVQ** queue attribute.

See [ALTER QUEUES](#) for more information on the **IMGRCOVQ** attribute.

Notes:

1. Media images are supported only if you are using linear logging. If you have enabled automatic media images, but are using circular logging, an error message is issued and the automatic media images attribute of the queue manager is disabled.
2. If you have enabled automatic media images, but have not specified a frequency, either minutes or megabytes of log, an error message is issued and no automatic media images are written.
3. You can manually record a media image using `rcdmqimg` when you have set **IMGSCHED(AUTO)**, if you want.

This enables you to take media images at a time that is suitable for your enterprise, for example, when your system is quiet. Automatic media imaging takes account of these manual media images, because taking a manual media image resets the interval and log length, before which the next automatic media image is taken.

4. The queue manager writes persistent messages only in media images, not nonpersistent messages. This can reduce the size of media images when migrating to later IBM MQ versions.

Deciding how to set **IMGLOGLN** and **IMGINTVL**

By default, **IMGLOGLN** is set to off for queue managers other than Native HA queue managers. (Native HA queue managers are created with **IMGLOGLN** set to the value of 25% of the available space on the volume where the recovery logs are to be written.)

By default, **IMGINTVL** is set to 60 minutes. The interval specified by **IMGINTVL** is honored when enough new work has been carried out on the queue manager to make it worthwhile recording a new image. Otherwise the taking of new images is delayed.

You can alter the values of **IMGLOGLN** and **IMGINTVL** to achieve the best solution for your configuration. Make **IMGLOGLN** and **IMGINTVL** large enough, so the queue manager is only spending a fraction of its time recording media images, but small enough so that:

- Damaged objects can be recovered in a reasonable amount of time, and
- Small enough so that your log fits on your disk without running out of space.

If you set **IMGLOGLN**, a good practice is to make **IMGLOGLN** many times the amount of data on your queues and many times the data rate of your workload. The larger you make **IMGLOGLN**, the less time your queue manager spends recording media images.

Similarly, if you set **IMGINTVL**, a good practice is to make **IMGINTVL** many times the amount of time the queue manager takes to record a media image. You can find out how long it takes to record a media image by recording one manually.

If you make **IMGLOGLN** and **IMGINTVL** too large, recovering a damaged object might take a very long time, because all the extents since the last media image have to be replayed.

Make **IMGLOGLN** and **IMGINTVL** small enough, so that the maximum time taken to recover a damaged object is acceptable to you.

Making **IMGLOGLN** and **IMGINTVL** very large, means that the log grows very large because media images are recorded so rarely.



Attention: Ensure that a log of this size fits comfortably on your log filesystem, as your workload will get backed out if the log filesystem fills completely.

You can set both **IMGINTVL** and **IMGLOGLN**. This might be useful to ensure that automatic media images are taken regularly during heavy workload (controlled by **IMGLOGLN**), but are still taken occasionally when the workload is very light (controlled by **IMGINTVL**).

IMGINTVL and **IMGLOGLN** are targets for the interval and log data length between which automatic media images are taken.

These attributes should not be seen as a fixed maximum or minimum. In fact, the queue manager might decide to schedule an automatic media image sooner, if the queue manager perceives that it is a really good time:

- Because the queue is empty, so taking the media image is the most efficient in terms of performance, and
- A media image has not been recorded for a while

On occasion the gap between automatic media images might be somewhat longer than either, or both, of **IMGINTVL** and **IMGLOGLN**.

The gap between media images might be larger than **IMGLOGLN** if the amount of data on queues is approaching **IMGLOGLN**. The gap between media images might be larger than **IMGINTVL** if it takes almost as long as **IMGINTVL** to record a media image.

This is poor practice because the queue manager would be spending much of its time recording media images.

When using automatic media image recording, the queue manager records a media image for each object and queue individually, so the queue manager tracks the interval and log length between images separately for each object.

Gradually over time, recording of media images becomes staggered, instead of recording media images for all objects at the same time. This staggering spreads out the performance impact of recording media images, and is another advantage of using automatic recording of media images over manual recording.

Taking media images manually - linear logging only

Recording a media image of a queue involves writing all the persistent messages from that queue to the log. For queues containing large volumes of message data, this involves writing a large amount of data to the log, and this process can impact the performance of the system while it is happening.

Recording media images of other objects is likely to be comparatively quick, since the media image of other objects does not contain user data.

You need to carefully consider when to record the media images of queues, so that the process does not interfere with your peak workload.

You must record the media image of all objects regularly, in order to update the oldest log extent needed for media recovery.

A good time to record the media image of a queue is when it is empty, because at that point no message data is written to the log. Conversely, a bad time is when the queue is very deep or has very large messages on it.

A good time to record the media image of a queue is when your system is quiet; whereas a bad time is during peak workload. If your workload is always quiet at midnight, for example, you might decide to record media images at midnight every night.

Staggering the recording of each of your queues can spread the performance impact out, and so lessen its effect. The longer it has been since you last recorded media images, the more important it becomes to record them, as the number of log extents required for media recovery is increasing.

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 AIX 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 754](#). **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](#).

Manual recording of media images with **rcdmqimg** to manage log space is not required, if you have opted to use linear logging with queue manager controlled automatic media imaging.

Note: Messages AMQ7467 and AMQ7468 can also be issued at the time of running the **rcdmqimg** command.

Partial media images

It is good practice to use IBM MQ messages only for data that is expected to be consumed in the near future, so that each message is on a queue for a relatively short amount of time.

Conversely, it is poor practice to use IBM MQ messages to store data long term like a database.

It is also good practice to ensure your queues are relatively shallow, and poor practice to have deep queues whose messages have been on the queue for a long time.

By following these guidelines, you enables the queue manager to optimize the performance of automatic recording of media images.

Recording the media image of an empty queue is very efficient (from a performance point of view) whereas taking the media image of a queue with a large amount of data on it is very inefficient, because all that data has to be written to the log in the media image.

For shallow queues with recently put messages on it, the queue manager can make a further optimization.

If all the messages currently on the queue were put in the recent past, the queue manager might be able to record the media image on behalf of a time (*recovery point*) just before all the messages were put, and so be able to record the image of the empty queue. This process is very low cost in terms of performance.

If all the messages that were on the queue at the recovery point have subsequently been got, these messages do not need to be recorded in the media image, as they are no longer on the queue.

This is called a *partial media image*. Then, in the unlikely event that the queue needs to be recovered, all logs records that relate to this queue since the last media image, will be replayed, so restoring all the recently put messages.

Even if there were a few messages on the queue at the recovery point, that are currently on the queue (and so have to be recorded in the partial media image) it is still more efficient to record this smaller partial media image, than a full media image of all messages.

Ensuring that messages remain on queues for a brief amount of time is likely to improve the performance of automatic recording of media images.

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.

Changing your queue manager log from linear to circular

You can change your queue manager log type from linear to circular by using the **migmqlog** command.

Before you begin

Decide whether you want to use linear or circular logging by reviewing [Types of logging](#).

Decide whether you want to change your log type in place, or move the log to a new location. When you move the log to a new location by using the **migmqlog** command, the log path in the `qm.ini` file is updated so that when you start your queue manager it uses the changed log. You can specify a new location by using the **-ld** option. If you are moving your queue manager from an old disk to a new Advanced Format disk, it might be convenient to use the **-ld** option.

Procedure

1. Login as a member of the `mqm` group.
2. Ensure that you have enough space to change your log. You must ensure that there is space for at least the configured primary log files, and one secondary log file.
3. If you have already not done so, stop the queue manager by using the **endmqm -w** command.
4. If you have not already done so, take a backup of your queue manager.

For more information, see [“Backing up queue manager data” on page 760](#), and [“Backing up configuration files after creating a queue manager” on page 15](#).

5. Run the **migmqlog** command:

- If you choose to change your log type without changing the location of the log, use the following command:

```
migmqlog -m QMgrName -lc
```

- If you choose to change your log type and move the log to a new location, use the following command:

```
migmqlog -m QMgrName -lc -ld NewLogLocation
```

where *NewLogLocation* is an absolute file path that specifies the new location for the log file. Do not use a relative file path with the **-ld** parameter.

For more information, see **migmqlog**.

If, for any reason, for instance due to a power outage, **migmqlog** command stops before it has completed the process, rerun the same **migmqlog** command on the partially changed logs to complete the changes.

Results

The command runs, and the queue manager log type is updated. Note that **migmqlog** might take some minutes to complete if your log is very large. However, the command outputs progress messages from time to time.

Related tasks

[“Changing your queue manager log from circular to linear” on page 747](#)

You can change your queue manager log type from circular to linear by using the **migmqlog** command.

Changing your queue manager log from circular to linear

You can change your queue manager log type from circular to linear by using the **migmqlog** command.

Before you begin

Decide whether you want to use linear or circular logging by reviewing [Types of logging](#).

Decide whether you want to change your log type in place, or move the log to a new location. When you move the log to a new location by using the **migmqlog** command, the log path in the qm.ini file is updated so that when you start your queue manager it uses the changed log. You can specify a new location by using the **-ld** option. If you are moving your queue manager from an old disk to a new Advanced Format disk, it might be convenient to use the **-ld** option.

About this task



Attention: After you have changed your log, a media image will not have been recorded when the queue manager starts. Plan how you intend to record media images, either automatically by setting the attributes:

- IMGSCHEd
- IMGINTVL
- IMGLOGLN
- IMGRCOVO
- IMGRCOVQ

in [ALTER QMGR](#), or manually by periodically running [rcdmqimg](#).

Procedure

1. Login as a member of the mqm group.
2. Ensure that you have enough space to change your log. The space that is used by a linear log increases continuously as data is logged.
3. If you have already not done so, stop the queue manager by using the **endmqm -w** command.
4. If you have not already done so, take a backup of your queue manager.
For more information, see [“Backing up queue manager data”](#) on page 760, and [“Backing up configuration files after creating a queue manager”](#) on page 15.
5. Run the **migmqlog** command. Note that **migmqlog** might take some minutes to complete if your log is very large. However, the command outputs progress messages from time to time.

- If you choose to change your log type without changing the location of the log, use the following command:

```
migmqlog -m QMgrName -ll
```

- If you choose to change your log type and move the log to a new location, use the following command:

```
migmqlog -m QMgrName -ll -ld NewLogLocation
```

where *NewLogLocation* is an absolute file path that specifies the new location for the log file. Do not use a relative file path with the **-ld** parameter.

For more information, see **migmqlog**.

If, for any reason, for instance due to a power outage, **migmqlog** command stops before it has completed the process, rerun the same **migmqlog** command on the partially changed logs to complete the changes.

6. Start the queue manager, and set the appropriate image recovery and queue attributes for your environment.
7. Consider when to record manual images for objects that are recoverable.

Related tasks

[“Changing your queue manager log from linear to circular”](#) on page 747

You can change your queue manager log type from linear to circular by using the **migmqllog** 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.
- Determine the name for archive notification, issue the command DISPLAY QMSTATUS ARCHLOG.

You can reduce the number of secondary log extents when using circular logging by issuing the command **RESET QMGR TYPE(REDUCELOG)**.

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.

Coldstart: What to do if log extents are missing or corrupt

If your enterprise loses some or all of the log extents needed for restart recovery, the queue manager will be unable to replay the recovery log and so fails to restart. If you require your queue manager to restart when the recovery log is corrupt in any way, at the expense of maintaining data integrity, it is possible to do so, although strongly discouraged. This process is known as *coldstarting* a queue manager.

Important: Coldstarting a queue manager should be considered only in exceptional circumstances and carries data integrity risks as described on this page. IBM suggests that you rebuild a queue manager, in preference to coldstarting, in response to corrupt data files.

If a coldstart is required for operational reasons, contact IBM Support to review the root cause of the issue. You should replace a coldstarted queue manager with a rebuilt queue manager at the earliest opportunity.

The effects of coldstart

On coldstart, the queue manager creates an empty recovery log and relies on the data in the queue files and other object files in their existing state. Because the data in the queue files can be inconsistent, messages might be lost, duplicated, corrupted, or inconsistent.

The queue manager stores the configuration of all the other persisted objects in the recovery log, as well as in object files. Other internal state data is also recorded in the recovery log as well, so on coldstart, internal state data is reset and all this other configuration data might be inaccurate.

The effects of coldstart are unpredictable and wide-ranging so you should avoid a coldstart unless absolutely necessary. After coldstarting, the information in the queue and object files can be so inconsistent that the queue manager will not restart at all.

If the queue manager does restart, there is no simple way of discovering what message data or configuration can be relied on and what cannot. Also, after a coldstart, queues might be damaged and so become completely unusable.

Additionally, if you can get from, or put to, a particular queue, the messages on it might be corrupt, missing, or duplicated. Transactions and channels might be stuck in-doubt. Even if your queue manager coldstarts successfully and the queues look intact, the unpredictable effects of the coldstart might not be realized until much later.

What to do if you need to coldstart

Performing a coldstart should not be considered standard operational practice, and IBM strongly discourages you from doing this. However, if you are in a position where you definitely need to coldstart a queue manager, contact [IBM MQ Support](#).

The process for coldstarting a queue manager used to be much more complicated for a linear queue manager than a circular one. In IBM MQ 9.1.3, the coldstart process has been much simplified, and does not involve copying or renaming log extents any more.

Contact IBM Support, who will give you a key which you pass to the **strmqm** command to coldstart a queue manager.



Attention: The coldstart command still carries the same risks of losing data integrity as a manual coldstart, and IBM strongly discourages you from doing this.

Eliminating future cold starts: a request

The **strmqm** command requires a key to coldstart, because IBM MQ wants you to contact IBM MQ Support if you need to coldstart, as IBM MQ is keen to understand how you got into this situation.

Clearly coldstart is something that is best avoided. IBM MQ has gone to considerable effort to make sure that you will not need to coldstart your queue manager, and IBM is keen to discover if there is anything more the product can do to alleviate having to coldstart.

Precautions to avoid a coldstart

The default logging method when creating a queue manager is circular logging. With circular logging you allow the queue manager a particular number of primary and secondary log extents of a given size. Create your log filesystem large enough to contain all the primary and secondary log extents, and you should never need to administer them.

Alternatively, you can use linear logging as opposed to circular. Linear logging gives you the added ability to recover queues and other objects, in the unlikely event that they become damaged. But by default, linear logging requires you to delete log extents that are no longer needed for restart or media recovery. This is referred to as manual log management.

When administering log extents in this way, it is possible to inadvertently delete too many log extents and so end up having to coldstart. To mitigate this risk, use automatic log management, so the queue manager manages log extents on your behalf.

The best practice is to put your recovery log in a separate log filesystem which only contains the recovery log. If you put your recovery log in the same filesystem as the rest of your queue manager, you can sometimes find that filesystem accidentally filling up, perhaps due to large queue files. Either make the log directory for the queue manager a separate filesystem, or specify a different log filesystem using the **-ld** command line option on the **crtmqm** command.

If the filesystem holding the queue files fills, you might not be able to put to those queues, but the queue manager continues running. If the filesystem containing the recovery log fills, the queue manager ends abruptly and will not restart until you free up some space.

Be careful not to delete log extents needed for restart recovery, otherwise you might find yourself needing to coldstart. Sometimes you might find that you need to coldstart because the disk failed that contains

their recovery log. Best practice is to put the recovery log on a replicated disk and so mitigate the risk of a disk crash.

Moving your messages and configuration to a new replacement queue manager avoids the possibility of ongoing problems with a queue manager that has been previously coldstarted.

Keep a note of which queue managers have been previously coldstarted, even if they were coldstarted a long time ago and have been stopped, restarted, and migrated in the meantime. When you contact IBM Support, say if the queue manager has been previously coldstarted and if so, give as much information as possible into what caused the requirement for a coldstart.

Using the log for recovery

You can use information from the logs to help you 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. It can also 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

You can record media images for objects so that they can be recovered if damaged. This feature is only available on queue managers that use linear logging or replicated logging and, for linear logging, only for objects that are defined as recoverable. You define that types of object are recoverable by using the **IMGRCOVO** and **IMGRCOVQ** queue manager attributes, see [ALTER QMGR](#). If an object that is not defined as recoverable is damaged, then the recovery options are the same as for circular logging.

Media recovery re-creates objects from information recorded in a linear log or replicated 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.

The queue manager records media images automatically if **IMGSCHED(AUTO)** is set. For more information, see [ALTER QMGR](#) for information on **IMGINTVL** and **INGLOGLN**.

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.

 Native HA queue managers use replicated logging. Such queue managers attempt automatic recovery of eligible objects when damage is detected. Once started, Native HA queue managers, by default, automatically attempt asynchronous recovery when object damage is detected. Recovery might not immediately be possible if, for example, the object is in use by an application, or the log extents required for media recovery are unavailable. In these situations, the asynchronous recovery processing periodically retries. If the issue that prevented recovery is resolved, the object will be recovered on the next retry, or the object can be recovered manually, using the **rcrmqobj** command.

What object files exist

The queue manager stores the attributes of objects that are defined in **runmqsc** in files on disk. These object files are in sub directories under the data directory of the queue manager.

  For example, on AIX and Linux platforms, channels are stored in `/var/mqm/qmgrs/qmgr/channel`.

The data in these object files is the media image of the objects. If these object files get deleted or corrupted, the object stored in that file is damaged. Using a linear logging queue manager, damaged

objects can be recovered from the log using the `rcrmqobj` command. Replicated logging (Native HA) queue managers automatically attempt to recover damaged objects when they are detected.

Most object files contain just the attributes of the object, so channel files contain the attributes of channels. The exceptions are:

- Catalog

The object catalog catalogs all the objects of all types and is stored in `qmanager1/QMQMOBJCAT`.

- Syncfiles

The syncfile contains internal state data associated with all channels.

- Queues

Queue files contain both the messages on that queue as well as the attributes of that queue.

Note that there is no catalog or syncfile object exposed in `runmqsc` or IBM MQ Explorer.

The catalog and the queue manager can be recorded, but not recovered. If these objects get damaged the queue manager ends preemptively and these objects get recovered automatically on restart.

Subscriptions are not listed in objects to record or recover, because durable subscriptions are stored on a system queue. To record or recover durable subscriptions, record or recover the `SYSTEM.DURABLE.SUBSCRIBER.QUEUE` instead.

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 (other than for Native HA queue managers, which use automatic recovery by default). At other times, when object damage is detected, operator message AMQ7472 is issued and most operations using the object fail with the return code `MQRC_OBJECT_DAMAGED`. If the queue manager object is damaged at any time after the queue manager has started, the queue manager performs a preemptive 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).

If a queue (or other object) gets damaged, **MEDIALOG** will not move forward. This is because **MEDIALOG** is the oldest extent required for media recovery. If your workload is continuing, **CURRLOG** will still be moving forward and so new extents will be written. Depending on your configuration (including your **LogManagement** setting), this might start filling your log filesystem. If the log filesystem fills completely, transactions get rolled back, and the queue manager might end abruptly. So when a queue gets damaged,

you might have only a limited amount of time to act before your queue manager ends. How much time you have, depends on the rate at which your workload is causing the queue manager to write new extents, and the amount of free space you have in your log filesystem.

If you are using manual log management, you might be archiving extents not needed for restart recovery, and then deleting them from the log filesystem, even though they are still needed for media recovery. This is acceptable as long as you can restore them from your archive when needed. This policy does not cause your log filesystem to fill when a queue gets damaged and **MEDIALOG** stops moving forward. However, if you only archive and delete extents that are not needed for either restart or media recovery, your log filesystem starts to fill if a queue gets damaged.

If you are using automatic or archive log management, the queue manager will not reuse extents that are still needed for media recovery, even though you might have archived them and notified the queue manager using `SET LOG ARCHIVED`. Consequently if a queue gets damaged your log filesystem will start filling.

If a queue gets damaged you will get OBJECT DAMAGED FFDCs written and **MEDIALOG** stops moving forward. The damaged object can be identified from the FFDC or because it is the object with the oldest **MEDIALOG** when you display its status in `runmqsc`.

If your log filesystem is filling, and you are concerned that your workload is getting backed out because the log filesystem is becoming full, then recovering the object, or quiescing your workload might stop this happening.

 In the case of Native HA queue managers, which use replicated logging, automatic recovery of damaged objects is attempted. Once started, Native HA queue managers, by default, automatically attempt asynchronous recovery when object damage is detected. Recovery might not immediately be possible if, for example, the object is in use by an application, or the log extents required for media recovery are unavailable. In these situations, the asynchronous recovery processing periodically retries. If the issue that prevented recovery is resolved, the object will be recovered on the next retry, or the object can be recovered manually, using the `rcrmqobj` command.

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:

<i>logactive</i>	The number of primary log extents.
<i>loginactive</i>	The number of secondary log extents.
<i>logsize</i>	The number of 4 KB pages per extent.
<i>baselsn</i>	The first LSN in the log extent containing the head of the log.
<i>nextlsn</i>	The LSN of the next log record to be written.
<i>headlsn</i>	The LSN of the log record at the head of the log.
<i>tailsn</i>	The LSN identifying the tail position of the log.
<i>lowtranlsn</i>	The first LSN of the oldest in-flight transaction.
<i>minbufflsn</i>	The first LSN to read from when restarting the queue manager.
<i>hflag1</i>	Whether the log is CIRCULAR or LOG RETAIN (linear). A queue manager that has LOG RETAIN set might also show LOG REPLICATE, which indicates a Native HA queue manager using log replication.
<i>HeadExtentID</i>	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:

<i>LSN</i>	The log sequence number.
<i>CreationTime</i>	The creation time of the log record.
<i>LogRecdType</i>	The type of the log record.
<i>XTranid</i>	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*.

<i>QueueName</i>	The queue associated with this log record (if any).
<i>Qid</i>	The unique internal identifier for the queue.
V 10.0.0 <i>ThisLSN</i>	The LSN of this log record.
V 10.0.0 <i>PrevLSN</i>	The LSN of the previous log record within the same transaction (if any).
V 10.0.0 <i>FirstLSN</i>	The LSN of the first log record in the transaction, if it can be found in the log, or empty otherwise.

Start Queue Manager

This logs that the queue manager has started.

<i>StartDate</i>	The date that the queue manager started.
<i>StartTime</i>	The time that the queue manager started.

Stop Queue Manager

This logs that the queue manager has stopped.

<i>StopDate</i>	The date that the queue manager stopped.
<i>StopTime</i>	The time that the queue manager stopped.
<i>ForceFlag</i>	The type of shutdown used.

Start Checkpoint

This denotes the start of a queue manager checkpoint.

V 10.0.0 <i>ChkPtCount</i>	The total number of checkpoints that have taken place on this queue manager. Note: The count starts from when the feature became available, at 9.4.2
<i>ChkPtOps</i>	The number of operations checkpointed on this queue manager since the last checkpoint.
<i>ChkPtSize</i>	The amount of data checkpointed on this queue manager since the last checkpoint, in MB, rounded up.
<i>ChkPtTotalOps</i>	The total number of operations that have been checkpointed on this queue manager. Note: The count starts from when the feature became available, at 9.4.2
<i>ChkPtTotalSize</i>	The total amount of data checkpointed on this queue manager. Note: The count starts from when the feature became available, at 9.4.2

V 10.0.0 For Native HA queue managers, these fields give an indication of how much work has been performed on the queue manager in a persistent way. If a queue manager in a Native HA group becomes partitioned these fields can indicate which queue manager has processed more data. (The cumulative values start from when the queue manager was migrated to a release supporting this feature.) See [Resolving a partitioned \(split brain\) problem](#).

End Checkpoint

This denotes the end of a queue manager checkpoint.

<i>ChkPtLSN</i>	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:

<i>MapIndex</i>	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 <i>Get Message</i> log record can be found containing the same <i>QueueName</i> and <i>MapIndex</i> . At this point the <i>MapIndex</i> identifier can be reused for a subsequent put message to that queue.
 <i>MsgId</i>	Message identifier. This field displays in the header part of a Put log entry, as well as in the data section.
 <i>CorrelId</i>	Correlation identifier. This field displays in the header part of a Put log entry, as well as in the data section.
 <i>GroupId</i>	Group identifier. This field displays in the header part of a Put log entry, as well as in the data section.
<i>Data</i>	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.

<i>Data</i>	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:

<i>MapIndex</i>	Identifies the message that was retrieved from the queue. The most recent <i>Put Message</i> log record containing the same <i>QueueName</i> and <i>MapIndex</i> identifies the message that was retrieved.
 <i>MsgId</i>	Message identifier. If the put of the message is also found in the log, this field displays in the header part of a Put log entry as well as in the data section.
 <i>CorrelId</i>	Correlation identifier. If the put of the message is also found in the log, this field displays in the header part of a Put log entry as well as in the data section.
 <i>GroupId</i>	Group identifier. If the put of the message is also found in the log, this field displays in the header part of a Put log entry as well as in the data section.
<i>QPriority</i>	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.

Roll back 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:

<i>XTranid</i>	The transaction identifier.
<i>FirstLSN</i>	The LSN of the first log record associated with the transaction.
<i>LastLSN</i>	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:

<i>RMName</i>	The name of the resource manager.
<i>RMID</i>	The resource manager identifier. This is also logged in subsequent <i>Transaction Prepared</i> log records that record global transactions in which the resource manager is participating.
<i>SwitchFile</i>	The switch load file for this resource manager.
<i>XAOpenString</i>	The XA open string for this resource manager.
<i>XACloseString</i>	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.

<i>ObjName</i>	The name of the object that was created.
<i>UserId</i>	The user ID performing the creation.

Delete Object

This logs the deletion of an IBM MQ object.

<i>ObjName</i>	The name of the object that was deleted.

Summary message

When **dmpmqlog** has output all the requested log it outputs a summary of the log indicating how many transactions were still active (between the dump start and end points), and how many puts and gets will be affected by the decision to commit or rollback those transactions. For example:

```
AMQ7714I: DMPMQLOG command has reached the end of the log.  
AMQ7738I: DMPMQLOG command found 5 active transactions with 10 put and 16 get log records  
between <0:0:0:14> and <0:0:0:9650>.  
AMQ7702I: DMPMQLOG command has finished successfully.
```

Backing up and restoring IBM MQ queue manager data

You can protect queue managers against possible corruption caused by hardware failures by backing up queue managers and queue manager data, by backing up the queue manager configuration only, and by using a backup queue manager.

About this task



CAUTION: You must take great care if you move a queue manager to a different operating system. See [Moving a queue manager to a different operating system](#) for more information.

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.

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 is unable to restart or to be recovered from the log. If you back up the queue manager configuration, you can re-create the queue manager and all of its objects from saved definitions.

To back up queue manager configuration, the queue manager must be running.

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 is 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.

Procedure

- To back up and restore queue manager data see:
 - [“Backing up queue manager data” on page 760.](#)
 - [“Restoring queue manager data” on page 761.](#)
- To back up and restore the queue manager configuration see:
 - [“Backing up queue manager configuration” on page 761](#)
 - [“Restoring queue manager configuration” on page 762](#)

- To create, update and start a backup queue manager, see [“Using a backup queue manager” on page 763](#).

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

Before starting to back up the queue manager, 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 are 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:

Procedure

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 configuration information in .ini files on Multiplatforms” on page 94](#).

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 726](#), and the configuration files as described in [“Initialization and configuration files” on page 260](#). Some of the directories might be empty, but you need them all to restore the backup at a later date.

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.

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: Before you upgrade to a later version of IBM MQ, take a backup of the `qm.ini` file and the registry entries. The queue manager information is stored in the `qm.ini` file and can be used to revert to a previous version.

Related tasks

[Stopping a queue manager](#)

[“Backing up configuration files after creating a queue manager” on page 15](#)

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

Restoring queue manager data

Follow these steps to restore a backup of a queue manager's data.

Before you begin

Before starting the backup, 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 384](#) and [Clustering: Availability, multi-instance, and disaster recovery](#) for more information.

Note: When you upgrade to a later version of IBM MQ, make sure that you 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.

Procedure

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.

Make sure that you have a log control file as well as the log files.

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.

4. Update the configuration information files.

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.

5. 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 AIX and Linux systems](#).

Results

If the data was backed up and restored correctly, the queue manager will now start.

Backing up queue manager configuration

Backing up queue manager configuration can help you to rebuild a queue manager from its definitions if both the queue manager configuration and log data is lost due to the hardware failure and the queue manager is unable to restart or to be recovered from the log.

About this task

 On AIX, Linux, and Windows, you can use the **dmpmqcfg** command to dump the configuration of an IBM MQ queue manager.

 On IBM i, you can use the Dump MQ Configuration (**DMPMQMCFG**) command to dump the configuration objects and authorities for a queue manager.

Procedure

1. Make sure that the queue manager is running.
2. Depending on your platform, use one of the following commands to back up the queue manager configuration:

- ▶ **ALW** On AIX, Linux, and Windows: Execute the Dump MQ Configuration command, **dmpmqcfig**, 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:

```
dmpmqcfig -m MYQMGR -a > /mq/backups/MYQMGR.mqsc
```

- ▶ **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 QMQNAME(MYQMGR) OUTPUT(*MQSC) EXPATTR(*ALL) TOFILE(QMQMSAMP/QMQSC)
TOMBR(MYQMGRDEF)
```

Related tasks

[“Restoring queue manager configuration” on page 762](#)

You can restore the configuration for a queue manager from a backup by first making sure that the queue manager is running and then running the appropriate command for your platform.

Related reference

[dmpmqcfig \(dump queue manager configuration\)](#)

[Dump MQ Configuration \(DMPMQMCFG\)](#)

▶ Multi Restoring queue manager configuration

You can restore the configuration for a queue manager from a backup by first making sure that the queue manager is running and then running the appropriate command for your platform.

About this task

▶ **ALW** On AIX, Linux, and Windows, you can use the **runmqsc** command to restore the configuration of an IBM MQ queue manager.

▶ **IBM i** On IBM i, you can use the **STRMQMMQSC** command to restore the configuration objects and authorities for a queue manager.

Procedure

1. Make sure that the queue manager is running.
Note that, if damage to the data and logs is unrecoverable by other means, the queue manager might have been re-created.
2. Depending on your platform, use one of the following commands to restore the queue manager configuration:
 - ▶ **ALW** On AIX, Linux, and Windows, run **runmqsc** against the queue manager, use standard input redirection to restore the definitions from a script file that is generated by the Dump MQ Configuration (**dmpmqcfig**) command (see [“Backing up queue manager configuration” on page 761](#)). For example:

```
runmqsc MYQMGR < /mq/backups/MYQMGR.mqsc
```

- ▶ **IBM i** On IBM i: Run **STRMQMMQSC** against the queue manager, and use the **SRCMBR** and **SRCFILE** parameters to restore the definitions from the physical file member that is generated

by the Dump MQ Configuration (**DMPMQMCFG**) command (see [“Backing up queue manager configuration”](#) on page 761). For example:

```
STRMQMMQSC MQMNAME(MYQMGR) SRCFILE(QMQMSAMP/QMQSC) SRCMBR(MYQMGR)
```

Related tasks

[“Backing up queue manager configuration”](#) on page 761

Backing up queue manager configuration can help you to rebuild a queue manager from its definitions if both the queue manager configuration and log data is lost due to the hardware failure and the queue manager is unable to restart or to be recovered from the log.

Related reference

[dmpmqcfg](#) (dump queue manager configuration)

[runmqsc](#) (run MQSC commands)

[Dump MQ Configuration \(DMPMQMCFG\)](#)

[Start IBM MQ Commands \(STRMQMMQSC\)](#)

Using a backup queue manager

An existing queue manager can have a dedicated backup queue manager for disaster recovery purposes.

About this task

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 764 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.

Important: Once the old backup queue manager has become the new active queue manager, for whatever reason, there is no longer a backup queue manager. This is effectively a form of asynchronous replication, and so the new active queue manager is expected to be logically some time behind the old active queue manager. As such, the old active queue manager no longer acts as a backup to the new active queue manager.

Procedure

- For information on using a backup queue manager, see the following topics:
 - [“Creating a backup queue manager”](#) on page 764
 - [“Updating a backup queue manager”](#) on page 764
 - [“Starting a backup queue manager”](#) on page 765

Related concepts

[“Logging: Making sure that messages are not lost”](#) on page 726

IBM MQ records all significant changes to the persistent data controlled by the queue manager in a recovery log.

Creating a backup queue manager

You create a backup queue manager as an inactive copy of the existing queue manager.

About this task

Important: You can only use a backup queue manager when using linear logging.

A 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.

Procedure

1. Create a backup queue manager for the existing queue manager using the control command **crtmqm**.
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 760](#).
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. Run the **strmqm** control command on the backup queue manager as shown in the following example:

```
strmqm -r BackupQMName
```

This command 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.

Related reference

[crtmqm \(create queue manager\)](#)

[strmqm \(start queue manager\)](#)

Updating a backup queue manager

To ensure that a backup queue manager remains an effective method for disaster recovery it must be updated regularly.

About this task

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 before you back it up.



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 is replayed.

Procedure

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 “2” on page 764. Copy only log extent files, the ones beginning with "S...".
4. Run the **strmqm** control command on the backup queue manager as shown in the following example:

```
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.

Related reference

[RESET QMGR](#)

[DISPLAY QMSTATUS](#)

[strmqm \(start queue manager\)](#)

Starting a backup queue manager

You can substitute a backup queue manager for an unrecoverable queue manager.

About this task

When restoring a backup of a queue manager in a cluster, see “[Recovering a cluster queue manager](#)” on page 384 and [Clustering: Availability, multi-instance, and disaster recovery](#) for more information.

If an unrecoverable queue manager has a dedicated backup queue manager, you can activate the backup queue manager in place of the unrecoverable queue manager.

When an unrecoverable queue manager is substituted with 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.

Note: Even though the queue manager data and log files are held in different directories, make sure that you 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. Even if it does start, your data is likely to be corrupt.

Procedure

1. Run the **strmqm** control command to activate the backup queue manager as shown in the following example:

```
strmqm -a BackupQMName
```

The backup queue manager is activated. Now that it is active, the backup queue manager can no longer be updated.

2. Run the **strmqm** control command to start the backup queue manager as shown in the following example:

```
strmqm BackupQMName
```

IBM MQ regards this as restart recovery, and uses 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.

3. Restart all channels.
4. Check the resulting directory structure to ensure that you have all the required directories.

For more information about IBM MQ directories and subdirectories, see [Planning file system support](#).

5. 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.

Results

If the data was backed up and restored correctly, the queue manager now starts.

Related tasks

[“Restarting stopped channels” on page 253](#)

When a channel goes into STOPPED state, you have to restart the channel manually.

Related reference

[strmqm \(start queue manager\)](#)

Multi

Changes to cluster error recovery on servers on Multiplatforms

The queue manager reruns operations that caused problems, until the problems are resolved. If, after five days, the problems are not resolved, the queue manager shuts down to prevent the cache becoming more out of date.

The queue manager reruns operations that caused problems, until the problems are resolved. If, after five days, the problems are not resolved, the queue manager shuts down to prevent the cache becoming more out of date. As the cache becomes more out of date, it causes a greater number of problems.

Every aspect of cluster management is handled for a queue manager by the local repository manager process, `amqrmfa`. The process runs on all queue managers, even if there are no cluster definitions.

IBM MQ, rather than stop the repository manager and keep going without it, the repository manager reruns failed operations. If the queue manager detects a problem with the repository manager, it follows one of two courses of action.

1. If the error does not compromise the operation of the queue manager, the queue manager writes a message to the error log. It reruns the failed operation every 10 minutes until the operation succeeds. By default, you have five days to deal with the error; failing which, the queue manager writes a message to the error log, and shuts down. You can postpone the five day shutdown.
2. If the error compromises the operation of the queue manager, the queue manager writes a message to the error log, and shuts down immediately.

An error that compromises the operation of the queue manager is an error that the queue manager has not been able to diagnose, or an error that might have unforeseeable consequences. This type of error often results in the queue manager writing an FFST file. Errors that compromise the operation of the queue manager might be caused by a bug in IBM MQ, or by an administrator, or a program, doing something unexpected, such as ending an IBM MQ process.

The point of the change in error recovery behavior is to limit the time the queue manager continues to run with a growing number of inconsistent cluster definitions. As the number of inconsistencies in cluster definitions grows, the chance of abnormal application behavior grows with it.

The default choice of shutting down the queue manager after five days is a compromise between limiting the number of inconsistencies and keeping the queue manager available until the problems are detected and resolved.

You can extend the time before the queue manager shuts down indefinitely, while you fix the problem or wait for a planned queue manager shutdown. The five-day stay keeps the queue manager running through a long weekend, giving you time to react to any problems or prolong the time before restarting the queue manager.

Corrective actions

You have a choice of actions to deal with the problems of cluster error recovery. The first choice is to monitor and fix the problem and the second to monitor and postpone fixing the problem.

1. Monitor the queue manager error log for the error messages [AMQ9448](#) and [AMQ5008](#), and fix the problem.

[AMQ9448](#) indicates that the repository manager has returned an error after running a command. This error marks the start of trying the command again every 10 minutes, and eventually stopping the queue manager after five days, unless you postpone the shutdown.

[AMQ5008](#) indicates that the queue manager was stopped because an IBM MQ process is missing. [AMQ5008](#) results from the repository manager stopping after five days. If the repository manager stops, the queue manager stops.

2. Monitor the queue manager error log for the error message [AMQ9448](#), and postpone fixing the problem.

If you disable getting messages from `SYSTEM.CLUSTER.COMMAND.QUEUE`, the repository manager stops trying to run commands, and continues indefinitely without processing any work. However, any handles that the repository manager holds to queues are released. Because the repository manager does not stop, the queue manager is not stopped after five days.

Run an MQSC command to disable getting messages from `SYSTEM.CLUSTER.COMMAND.QUEUE`:
`ALTER QLOCAL(SYSTEM.CLUSTER.COMMAND.QUEUE) GET(DISABLED)`

To resume receiving messages from `SYSTEM.CLUSTER.COMMAND.QUEUE` run an MQSC command:
`ALTER QLOCAL(SYSTEM.CLUSTER.COMMAND.QUEUE) GET(ENABLED)`

Special consideration

Stopping `amqrmfa` in IBM MQ causes the queue manager to stop, because it is regarded as a queue manager failure. You must not stop the `amqrmfa` process unless you set the queue manager tuning parameter, `TolerateRepositoryFailure`.

Example

```
TuningParameters:
  TolerateRepositoryFailure=TRUE
```

Figure 86. Set `TolerateRepositoryFailure` to `TRUE` in `qm.ini`

Related concepts

[“Queue manager configuration files, `qm.ini`” on page 108](#)

A queue manager configuration file, `qm.ini`, contains information relevant to a specific queue manager. The attributes that you can use to modify the configuration of an individual queue manager override any settings for IBM MQ.

Configuring JMS and Jakarta Messaging resources

One of the ways in which a JMS or Jakarta Messaging 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

JM 3.0 From IBM MQ 9.3.0, Jakarta Messaging 3.0 is supported for developing new applications. IBM MQ 9.3.0 and later continue to support JMS 2.0 for existing applications. It is not supported to use both the Jakarta Messaging 3.0 API and the JMS 2.0 API in the same application. For more information, see [Using IBM MQ classes for JMS/Jakarta Messaging](#).

You can create and configure administered objects in IBM MQ by using either of the following tools:

IBM MQ JMS and Jakarta Messaging administration tools

The IBM MQ JMS administration tool, **JMSAdmin**, and Jakarta Messaging administration tool, **JMS30Admin**, are command-line tools that you can use to create and configure IBM MQ JMS and Jakarta Messaging objects that are stored in LDAP, in a local file system, or other locations. The JMS and Jakarta Messaging administration tools use a syntax that is similar to **runmqsc**, and also support scripting.

The administration tools use 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 JMSAdmin and JMS30Admin tools” on page 775](#).

> **JMS 2.0** IBM MQ Explorer

For JMS 2.0, you can use IBM MQ Explorer to create and administer JMS 2.0 object definitions that are stored in LDAP, in a local file system, or other locations.

> **JM 3.0** For Jakarta Messaging 3.0, you cannot administer JNDI using IBM MQ Explorer. JNDI administration is supported by the Jakarta Messaging 3.0 variant of **JMSAdmin**, which is **JMS30Admin**.

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.

> **JM 3.0** Although IBM MQ 9.3 and later support [Jakarta Messaging 3.0](#), WebSphere Application Server does not currently have equivalent support. Therefore, in WebSphere Application Server, you configure Java Message Service 2.0 resources.

IBM 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 concepts

[Creating and configuring connection factories and destinations in an IBM MQ classes for JMS application](#)

Related reference

[runmqsc \(run MQSC commands\)](#)

Configuring connection factories and destinations in a JNDI namespace

JMS and Jakarta Messaging applications access administered objects in the naming and directory service through the Java Naming and Directory Interface (JNDI). The JMS or Jakarta Messaging administered objects are stored in a location within the naming and directory service that is referred to as the JNDI

namespace. A JMS or Jakarta Messaging application can look up the administered objects to connect to IBM MQ and access destinations for sending or receiving messages.

About this task

JMS or Jakarta Messaging applications look up the names of the JMS or Jakarta Messaging 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 or Jakarta Messaging 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.

Before an IBM MQ classes for JMS or IBM MQ classes for Jakarta Messaging application can retrieve administered objects from a JNDI namespace, you must first create the administered objects. You can create and configure the following types of JMS or Jakarta Messaging object:

Connection factory

A JMS or Jakarta Messaging connection factory object defines a set of standard configuration properties for connections. A JMS or Jakarta Messaging 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. For more information, see [The JMS and Jakarta Messaging model](#).

Destination

A JMS or Jakarta Messaging 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 or Jakarta Messaging 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 or Jakarta Messaging queue destination used in point-to-point messaging
- JMS or Jakarta Messaging topic destination used in publish/subscribe messaging

► **JMS 2.0** For JMS 2.0, you can create contexts and administered objects by using either IBM MQ Explorer or the IBM MQ JMS administration tool **JMSAdmin**.

Note: ► **JM 3.0** For Jakarta Messaging 3.0, you cannot administer JNDI using IBM MQ Explorer. JNDI administration is supported by the Jakarta Messaging 3.0 variant of **JMSAdmin**, which is **JMS30Admin**.

The following diagram shows an example of JMS or Jakarta Messaging objects created in an IBM MQ JNDI namespace.

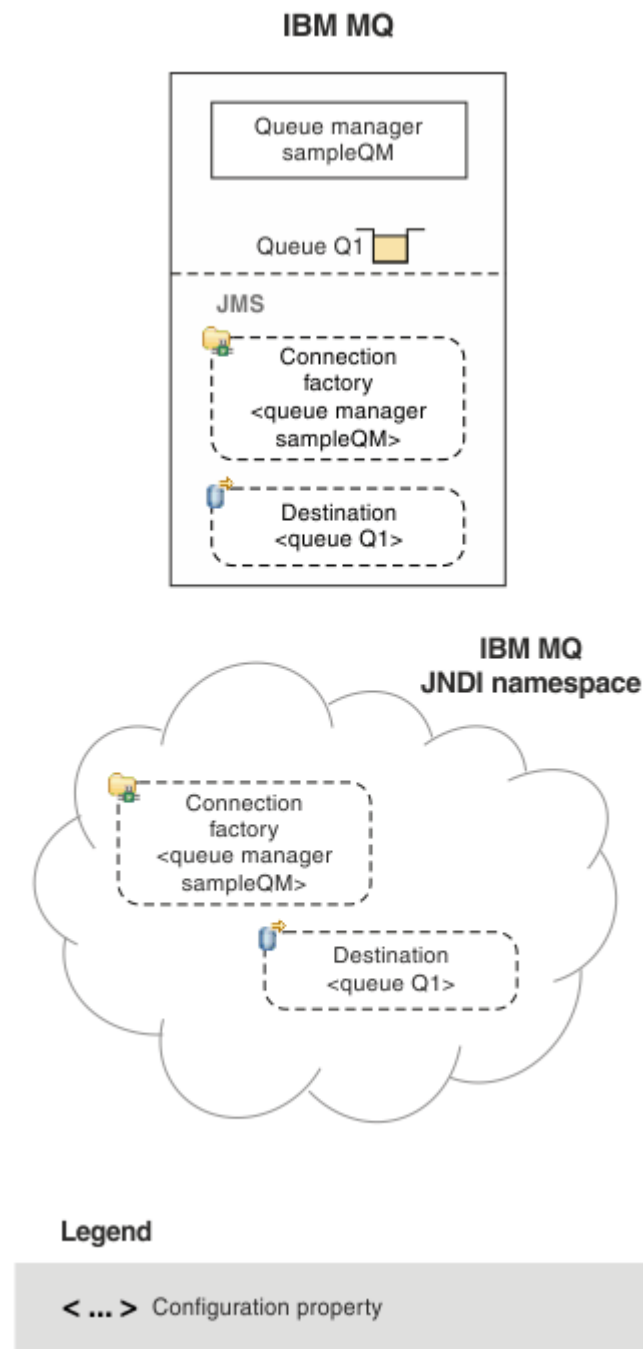


Figure 87. JMS or Jakarta Messaging objects created in IBM MQ

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.

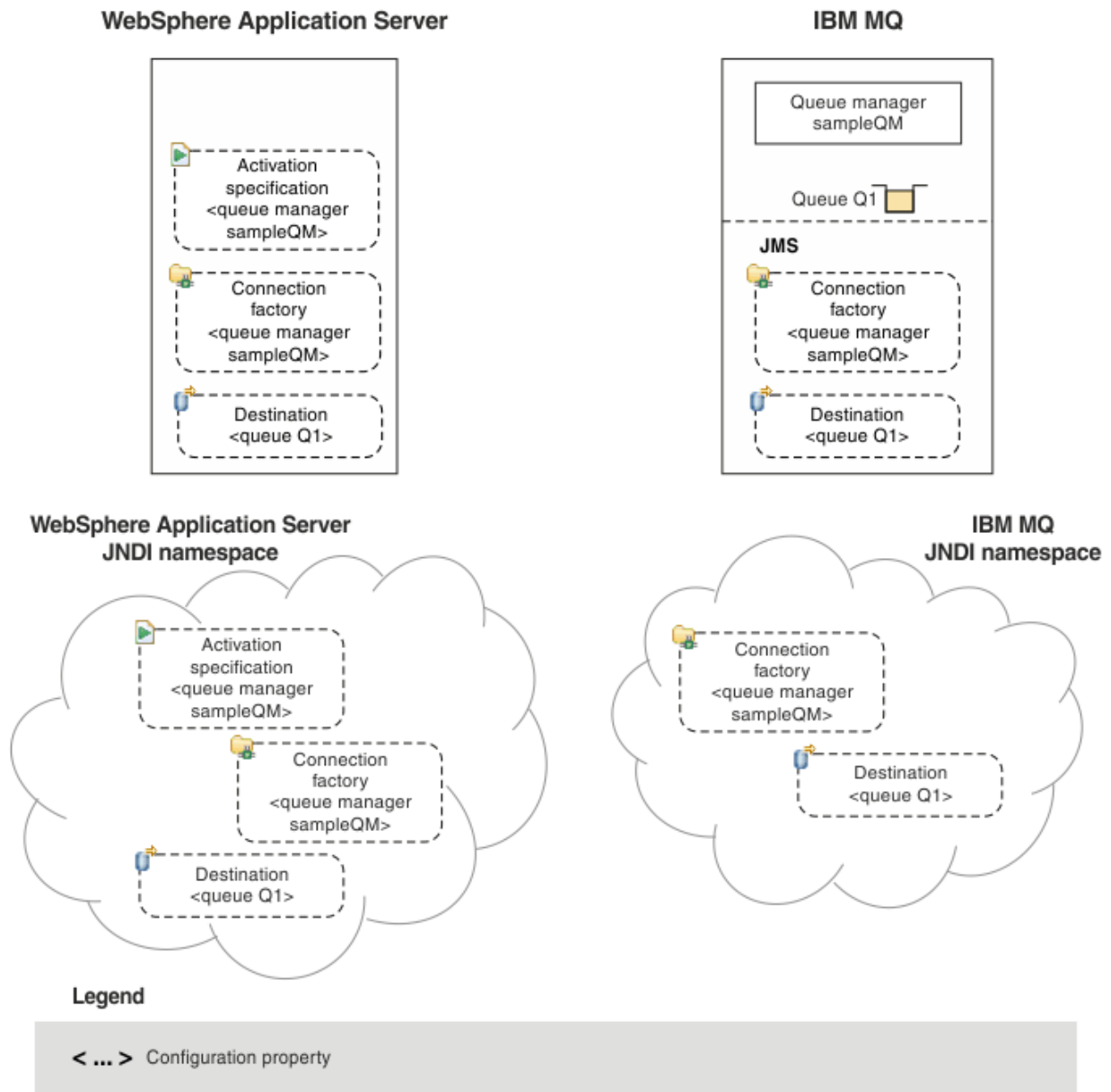


Figure 88. Objects created in WebSphere Application Server, and the corresponding objects in IBM MQ

JM 3.0 Although IBM MQ 9.3 and later support Jakarta Messaging 3.0, WebSphere Application Server does not currently have equivalent support. Therefore, in WebSphere Application Server, you configure Java Message Service 2.0 resources.

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

- ▶ **JMS 2.0**
To configure JMS objects for IBM MQ by using IBM MQ Explorer, see [“Configuring JMS 2.0 objects using IBM MQ Explorer”](#) on page 772.
- ▶ **JMS 2.0**
To configure JMS objects for IBM MQ by using the IBM MQ JMS administration tool, **JMSAdmin**, see [“Configuring JMS and Jakarta Messaging objects using the administration tools”](#) on page 773.
- ▶ **JM 3.0**
To configure Jakarta Messaging objects for IBM MQ by using the IBM MQ Jakarta Messaging administration tool, **JMS30Admin**, see [“Configuring JMS and Jakarta Messaging objects using the administration tools”](#) on page 773.
- ▶ **JMS 2.0**
To configure JMS objects for WebSphere Application Server, see [“Configuring JMS 2.0 resources in WebSphere Application Server”](#) on page 784.

Results

An IBM MQ classes for JMS or IBM MQ classes for Jakarta Messaging 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](#)

▶ **JMS 2.0** Configuring JMS 2.0 objects using IBM MQ Explorer

Use the IBM 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

▶ **JMS 2.0** IBM 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. IBM MQ Explorer runs on Windows and Linux for 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.

Note: ▶ **JM 3.0** For Jakarta Messaging 3.0, you cannot administer JNDI using IBM MQ Explorer. JNDI administration is supported by the Jakarta Messaging 3.0 variant of **JMSAdmin**, which is **JMS30Admin**.

In IBM 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 IBM MQ Explorer, either contextually from an existing object in the IBM 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 IBM 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 IBM MQ Explorer, if it is not already running.
If IBM 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 IBM 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 IBM 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 IBM MQ Explorer](#).

Related concepts

[Introduction to IBM MQ Explorer](#)

[Creating and configuring connection factories and destinations in an IBM MQ classes for JMS application](#)

Configuring JMS and Jakarta Messaging objects using the administration tools

IBM MQ provides administration tools that you can use to define the properties of eight types of IBM MQ classes for JMS or IBM MQ classes for Jakarta Messaging 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

JMS 2.0 For JMS 2.0, JNDI administration is supported by the **JMSAdmin** tool.

JM 3.0 For Jakarta Messaging 3.0, JNDI administration is supported by the Jakarta Messaging 3.0 variant of **JMSAdmin**, which is **JMS30Admin**.

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 37 on page 773](#).

Table 37. The JMS and Jakarta Messaging 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.

Table 37. The JMS and Jakarta Messaging object types that are handled by the administration tool (continued)

Object Type	Keyword	Description
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.
MQXAConnectionFactory “1” on page 774	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 774	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 774	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 783](#).

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 779](#).

JMS 2.0 For JMS 2.0 and earlier, you can also create and configure IBM MQ classes for JMS administered objects with IBM MQ Explorer.

JM 3.0 For Jakarta Messaging 3.0, you cannot administer JNDI using IBM MQ Explorer. JNDI administration is supported by the Jakarta Messaging 3.0 variant of **JMSAdmin**, which is **JMS30Admin**.

Related concepts

[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 JMSAdmin and JMS30Admin tools

The IBM MQ JMS and Jakarta Messaging administration tools use a configuration file to set the values of certain properties. In each case, a sample configuration file is supplied that you can edit to suit your system.

About this task

JM 3.0 IBM MQ 9.3.0 introduced support for [Jakarta Messaging 3.0](#). JMS 2.0 is still fully supported.

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` (for JMS 2.0) or `JMS30Admin.config` (for Jakarta Messaging 3.0). This file is found in the `MQ_JAVA_INSTALL_PATH/bin` directory. You can either edit the 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 JMSAdmin and JMS30Admin tools” on page 777](#). 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` or `JMS30Admin.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 or IBM MQ classes for Jakarta Messaging installation.

The names of JMS or Jakarta Messaging 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)
-  `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 or JMS30Admin 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 JMSAdmin and JMS30Admin tools

The IBM MQ JMS and Jakarta Messaging administration tools have 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:

```
> JMS 2.0
```

```
JMSAdmin [-t] [-v] [-cfg config_filename] [-sf keyfile]
```

```
> JM 3.0
```

```
JMS30Admin [-t] [-v] [-cfg config_filename] [-sf keyfile]
```

where:

-t

Enables trace (default is trace off).

The trace file is generated in "%MQ_JAVA_DATA_PATH%\errors (Windows) or /var/mqm/trace (AIX and Linux). 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 (for JMS 2.0) or JMS30Admin.config (for Jakarta Messaging 3.0), is used. For more information about the configuration file, see [“Configuring the JMSAdmin and JMS30Admin tools”](#) on page 775.

-sf: Optional

Defines the absolute path of the file containing a user supplied key used for encrypting the token secret. A default key is used if this parameter is not specified. For more information about creating a connection factory with JWT token properties, see [Using authentication tokens in an application](#).

A command prompt is displayed, which indicates that the tool is ready to accept administration commands. This prompt initially appears as:

```
InitCtx>
```

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 JMSAdmin and JMS30Admin tools” on page 775](#).

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:

```
JMS 2.0
```

```
JMSAdmin < test.scf
```

```
JM 3.0
```

```
JMS30Admin < test.scf
```

where *test.scf* is a script file that contains administration commands. For more information, see [“Using administration commands with JMSAdmin and JMS30Admin” on page 778](#). The last command in the file must be the END command.

Using administration commands with JMSAdmin and JMS30Admin

The IBM MQ JMS and Jakarta Messaging administration tools accept 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 tools.

Table 38. 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
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 JMSAdmin and JMS30Admin tools” on page 777](#).

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 779](#).

- Enter commands in the following form:

```
verb [param]*
```

where **verb** is one of the administration verbs listed in [Table 38 on page 778](#). 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) +  
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 39. 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 or Jakarta Messaging 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 JMSAdmin and JMS30Admin tools” on page 775](#).

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 or Jakarta Messaging 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 or Jakarta Messaging 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 JMSAdmin and JMS30Admin tools” on page 775](#).

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 JMSAdmin and JMS30Admin tools” on page 777](#).

The command prompt is displayed, indicating that the tool is ready to accept administration commands.

2. Make sure that the 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 779](#).

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

Example: Creating a queue

The following examples show a queue called testQueue created in the initial context by 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.

JM 3.0

```
InitCtx> DEFINE Q(testQueue)

InitCtx> DISPLAY CTX

Contents of InitCtx

a cn=testQueue          com.ibm.mq.jakarta.jms.MQQueue

1 Object(s)
0 Context(s)
1 Binding(s), 1 Administered
```

JMS 2.0

```
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
```

Example: creating a connection factory with JWT token properties

The following examples show a connection factory called CONFACT created in the initial context using the DEFINE verb.

- Example that uses the long names of the **TOKEN_CLIENT_ID** and **TOKEN_ENDPOINT** properties:

```
DEFINE CF(CONFACT) HOSTNAME(localhost) PORT(1414) CHANNEL(QM1_SVRCONN)
TOKENCLIENTID(MyClientID)
TOKENENDPOINT(https://keycloak.ibm.com:8080/api/token)
```

- Example that uses the short names of the **TOKEN_CLIENT_ID** and **TOKEN_ENDPOINT** properties.

```
DEFINE CF(CONFACT) HOST(localhost) PORT(1414) HOSTNAME(localhost) PORT(1414)
CHANNEL(QM1_SVRCONN)
TCID(MyClientID) TKEP(https://keycloak.ibm.com:8080/api/token)
```

As the JMSAdmin or JMS30Admin tool identifies a token endpoint has been provided, it prompts the user to enter the token secret. The console reads the user input with echoing disabled. The token secret is encrypted before it is written to the bindings file.

Note: Since the token secret is a confidential attribute, if it is provided while defining the connection factory properties, the appropriate error is displayed in order to prevent someone from reading the token secret inadvertently.

The following example shows how to display the connection factory properties on the JMSAdmin or JMS30Admin tool:

```
InitCtx> DISPLAY CF(CONFACT)

CHANNEL(CHAN1)
CONNECTIONNAMELIST(9.30.42.223(1513))
HOSTNAME(9.30.42.223)
PORT(1513)
QMANAGER(DEMO)
TEMPTOPICPREFIX()
TOKENCLIENTID(jms-client)
TOKENENDPOINT(https://9.30.42.223:32030/realms/master/protocol/openid-connect/token)
TOKENSECRET(*****)
TRANSPORT(CLIENT)

InitCtx>
```

Note: In this example, the **TOKEN_CLIENT_SECRET** property is shown as asterisk characters (*****) because it is confidential.

Example: creating a connection factory when supplying a customized SSLSocketFactory

The following examples show the usage of properties in defining a connection factory called CONFACTJWT or CONFACTCCDT when an application supplies a customized SSLSocketFactory when retrieving a JWT or CCDTURL from an HTTPS endpoint.

- Example using long and short names of **TOKEN_HTTPS_CERT_VALIDATION_POLICY**:

```
DEFINE CF(CONFACTJWT) HOSTNAME(localhost) PORT(1414) CHANNEL(QM1_SVRCONN)
TOKENCLIENTID(MyClientID) TOKENENDPOINT(https://keycloak.ibm.com:8080/api/token)
TOKENCERTVALPOL(ANY)
```

```
DEFINE CF(CONFACTJWT) HOST(localhost) PORT(1414) CHAN(QM1_SVRCONN) TKCD(MyClientID)
TKEP(https://keycloak.ibm.com:8080/api/token) TCVP(ANY)
```

- Example using long and short names of **WMQ_CCDT_HTTPS_CERT_VALIDATION_POLICY**:

```
DEFINE CF(CONFACTCCDT) CCDTURL(https://ccdturl.com/CCDT.json) QMANAGER(R93)
CCDTCERTVALPOL(ANY) TRANSPORT(CLIENT)
```

```
DEFINE CF(CONFACTCCDT) CCDT(https://ccdturl.com/CCDT.json) QMGR(R93) CCVP(ANY)
TRAN(CLIENT)
```

For more information, see [Supplying a customized SSLSocketFactory in IBM MQ classes for JMS/Jakarta Messaging](#).

Related reference

[Properties of IBM MQ classes for JMS objects](#)

Sample error conditions creating a JMS object

A number of common error conditions can arise when you create an object.

Here 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
```

Here 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 40 on page 783 summarizes the use of these verbs. Substitute *TYPE* with the keyword that represents the required administered object, as described in [“Configuring JMS and Jakarta Messaging objects using the administration tools”](#) on page 773.

Table 40. Syntax and description of commands used to manipulate administered objects	
Command syntax	Description
<code>ALTER TYPE (name) [property]*</code>	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.
<code>DEFINE TYPE (name) [property]*</code>	Attempts to create an administered object of type <i>TYPE</i> with the supplied properties, and store it under the name <i>name</i> 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.

Table 40. Syntax and description of commands used to manipulate administered objects (continued)

Command syntax	Description
DISPLAY <i>TYPE</i> (name)	Displays the properties of the administered object of type <i>TYPE</i> , bound under the name name in the current context. Fails if the object does not exist, or if there is a security violation.
DELETE <i>TYPE</i> (name)	Attempts to remove the administered object of type <i>TYPE</i> , having the name name, from the current context. Fails if the object does not exist, or if there is a security violation.
COPY <i>TYPE</i> (nameA) <i>TYPE</i> (nameB)	Makes a copy of the administered object of type <i>TYPE</i> , 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 <i>TYPE</i> (nameA) <i>TYPE</i> (nameB)	Moves (renames) the administered object of type <i>TYPE</i> , 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 2.0 resources in WebSphere Application Server

To configure JMS 2.0 resources in WebSphere Application Server, you can either use the administrative console or wsadmin commands.

Before you begin

 Although IBM MQ 9.3 and later support [Jakarta Messaging 3.0](#), WebSphere Application Server does not currently have equivalent support. Therefore, in WebSphere Application Server, you configure Java Message Service 2.0 resources.

About this task

Java Message Service 2.0 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 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 and later, 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 tasks

[Using IBM MQ and WebSphere Application Server together](#)

WebSphere Application Server topics

[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 2.0 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 IBM 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

[“Multi-instance queue managers” on page 540](#)

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.

Related tasks

[“Configuring a binary format CCDT” on page 43](#)

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On Multiplatforms, a binary CCDT containing default settings is created automatically when the queue manager is created. You use the **runmqsc** command to update a binary CCDT.

[“Configuring publish/subscribe messaging” on page 444](#)

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.

WebSphere Application Server topics

[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 2.0 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 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 **listWMQConnectionFactory** 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 **showWMQTopic** 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

[“Multi-instance queue managers” on page 540](#)

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.

Related tasks

[“Configuring a binary format CCDT” on page 43](#)

The client channel definition table (CCDT) determines the channel definitions and authentication information used by client applications to connect to the queue manager. On Multiplatforms, a binary CCDT containing default settings is created automatically when the queue manager is created. You use the **runmqsc** command to update a binary CCDT.

[“Configuring publish/subscribe messaging” on page 444](#)

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.

WebSphere Application Server topics

[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](#)

JMS 2.0 Using JMS 2.0 shared subscriptions

In WebSphere Application Server 9.0, you can configure and use JMS 2.0 shared subscriptions with IBM MQ 9.0.

About this task

The JMS 2.0 specification introduced the concept of shared subscriptions, which enables a single subscription to be opened by one or more consumers. The messages are shared amongst all of these consumers. There is no restriction where these consumers are so long as they connect to the same queue manager.

Shared Subscriptions can be either durable or non-durable, with the same semantics as what are now referred to as unshared subscriptions.

For a consumer to be able to identify which subscription to use, it needs to supply a subscription name. This is similar to unshared durable subscriptions, but a subscription name is required in all cases where a shared subscription is required. A clientID, however, is not required in the case of a durable shared-subscription; one can be supplied but it is not mandatory.

Whilst shared subscriptions can be thought of as a loading balancing mechanism, neither in IBM MQ nor the JMS 2.0 specification is there any commitment as to how the messages are distributed amongst consumers.

In WebSphere Application Server 9.0 an IBM MQ 9.0 resource adapter is pre-installed.

The following steps show how to configure an activation specification to use a shared durable or a shared non-durable subscription using the WebSphere Application Server administrative console.

Procedure

First create the objects in JNDI.

1. Create a topic destination in JNDI as normal (see [“Configuring JMS 2.0 resources using the administrative console”](#) on page 785).
2. Create the activation specification (see [“Configuring JMS 2.0 resources using the administrative console”](#) on page 785).

You can create the activation specification with exactly the properties that you need. If you want to use a durable subscription, you can select it on creation and specify a name. If you want to use a non-durable subscription, you cannot specify a name at this point. Instead, you need to create a custom property for the subscription name.

Update the activation specification that you created with the required custom properties. There are two custom properties that you might need to specify:

- In all cases, you must create a custom property to specify that this activation specification should use a shared subscription.
- If the subscription was created as non-durable, the subscription name property needs to be set as a custom property.

The following table shows the valid value that you can specify for each custom property:

Property Name	Type	Valid values
sharedSubscription	String	true, false
subscriptionName	String	Non-zero length java String

3. Select the activation specification from the list displayed in the **Activation specification collection** form.
The details for the activation specification are displayed in the **IBM MQ messaging provider activation specification settings** form.
4. On the **IBM MQ messaging provider activation specification settings** form, click **Custom properties**.
The **Custom properties** form is displayed.
5. If you are using a non-durable subscription, create the subscriptionName custom property.
On the **Custom properties** panel of the activation specification, click **New**, then enter the following details:

Name

The name of the custom property, which in this case is subscriptionName.

Value

The value for the custom property. You could use the JNDI names in the **Value** field , for example WASSharedSubOne.

Type

The type of the custom property. Select the custom property type from the list, which in this case must be `java.lang.String`.

6. For both a shared durable and shared non-durable subscription, create the `sharedSubscription` custom property.

On the **Custom properties** panel of the activation specification, click **New**, then enter the following details:

Name

The name of the custom property, which in this case is `sharedSubscription`.

Value

The value for the custom property. To specify that the activation specification uses a shared subscription, set the value to `true`. If you later wanted to stop using a shared subscription for this activation specification, you could do this by setting the value of this custom property to `false`.

Type

The type of the custom property. Select the custom property type from the list, which in this case must be `java.lang.String`.

7. When the properties are set, restart the application server.

The message-driven beans (MDB)s for the activation specifications are then driven when messages arrive, but only the MDBs share the messages that are sent.

Related concepts

[Cloned and shared subscriptions](#)

[Subscription durability](#)

Related tasks

[Configuring the resource adapter for inbound communication](#)

Related information for WebSphere Application Server 9.0

[Configuring a topic for the IBM MQ messaging provider](#)

[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](#)

[Configuring custom properties for IBM MQ messaging provider JMS resources](#)

Using JMS 2.0 ConnectionFactory and Destination Lookup properties

In WebSphere Application Server 9.0, the `ConnectionFactoryLookup` and `DestinationLookup` properties of an activation specification can be provided with a JNDI name of an administered object to be used in preference to the other activation specification properties.

About this task

The JMS 2.0 specification specifies two additional properties on the activation specification used to drive message-driven beans (MDBs). Previously, each vendor had to specify custom properties on the activation specification to provide details that are required to connect to a messaging system and to define which destination to get messages from.

The now standard `connectionFactoryLookup` and `destinationLookup` properties can be used to give a JNDI name of the relevant object to look up and use. Within WebSphere Application Server 9.0 an IBM MQ 9.0 resource adapter is pre-installed.

The following steps show how to customize and use these two properties using the WebSphere Application Server administrative console.

Procedure

First create the objects in JNDI.

1. Create the `ConnectionFactory` in JNDI as normal (see [“Configuring JMS 2.0 resources using the administrative console” on page 785](#)).
2. Create the `Destination` in JNDI as normal (see [“Configuring JMS 2.0 resources using the administrative console” on page 785](#)).

The `Destination` object must have the correct values.

3. Create the activation specification using any values that are needed (see [“Configuring JMS 2.0 resources using the administrative console” on page 785](#)).

You can create the activation specification with exactly the properties that you need. However, you should bear in mind the following considerations:

- If you want the IBM MQ resource adapter to use the Java EE connection factory and destination lookup properties, it is less relevant what properties are used when you create the activation specification (see [ActivationSpec ConnectionFactoryLookup and DestinationLookup properties](#)).
- However, any property that is not already defined on either the connection factory or the destination must still be specified on the activation specification. Therefore, you must define the connection consumer properties and additional properties, and the authentication information that is used when a connection is actually created.
- Of the properties that are defined on the connection factory, the `ClientID` property has special processing. This is because a common scenario is using a single connection factory with multiple activation specifications. This simplifies administration, however the JMS specification does demand unique client ids, hence the activation specification needs to have the ability to override any value set on the `ConnectionFactory`. If no `ClientID` is set on the activation specification, any value on the connection factory is used.

Either update the activation specification that you have created with the two new custom properties by using the WebSphere Application Server administrative console as described in step [“4” on page 792](#), or use annotations instead as described in step [“5” on page 793](#).

4. Update the activation specification in the WebSphere Application Server administrative console.

These two properties need to be set on the custom properties panel of the activation specification. These properties are not present in the main activation specification panels or on the Activation Specification creation wizard.

- a) Select the activation specification from the list displayed in the **Activation specification collection** form.

The details for the activation specification are displayed in the **IBM MQ messaging provider activation specification settings** form.

- b) On the **IBM MQ messaging provider activation specification settings** form, click **Custom properties**.

The **Custom properties** form is displayed.

- c) On the **Custom properties** form, create two new custom properties, both of type `java.lang.String`. In each case, click **New** and then enter the following details for the custom property:

Name

The name of the custom property, either `connectionFactoryLookup` or `destinationLookup`.

Value

The value for the custom property. You could use the JNDI names in the **Value** field , for example QuoteCF and QuoteQ.

Type

The type of the custom property. Select the custom property type from the list, which in this case must be `java.lang.String`.

The deployed MDB will now use these values to create the connection factory and destination. When deploying the MDB, there is no requirement to set the JNDI value configuration.

5. Use annotations instead of the activation specification.

It is possible to use annotations in the MDB code to specify values as well. For example, using the JNDI names QuoteCF and QuoteQ, this is what the code would look like:

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "destinationType" , propertyValue =
"javax.jms.Topic" ),
    @ActivationConfigProperty(propertyName = "destinationLookup" , propertyValue =
"QuoteQ" ),
    @ActivationConfigProperty(propertyName = "connectionFactoryLookup" , propertyValue
= "QuoteCF" )}, mappedName = "LookupMDB" )
@TransactionAttribute(TransactionAttributeType.REQUIRED)
@TransactionManagement(TransactionManagementType.CONTAINER)
publicclass LookupMDB implements MessageListener {
```

Related tasks

[Configuring the resource adapter for inbound communication](#)

Related information for WebSphere Application Server 9.0

[Configuring a unified connection factory for the IBM MQ messaging provider](#)

[Configuring a topic for the IBM MQ messaging provider](#)

[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](#)

[Configuring custom properties for IBM MQ messaging provider JMS resources](#)

Configuring WebSphere 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:

-  WebSphere Application Server does not currently support Jakarta EE. See [IBM MQ resource adapter statement of support](#).
- If you are using WebSphere Application Server 8.5 or earlier on any platform, do not install the IBM MQ 8.0 or later resource adapter into the application server. The IBM MQ 8.0 or later resource adapter can only be deployed into an application server that supports JMS 2.0. However, WebSphere Application Server 8.5 or earlier only supports JMS 1.1.

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 7.0.0 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 7.0.1 Fix Pack 5, 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 fix pack. 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 WebSphere Application Server 7.0.1 Fix Pack 5.

```
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 MQ Resource Adapter") or (desc == "WAS 7.0.0.1 Built In MQ Resource Adapter"):
        print "Updating archivePath and classpath of " + ra
        AdminConfig.modify(ra, [['archivePath', "${WAS_INSTALL_ROOT}/installedConnectors/wmq.jmsra.rar"]])
        AdminConfig.unsetAttribute(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 concepts

[Using the IBM MQ resource adapter](#)

Related information for WebSphere Application Server 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](#)

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 an IBM MQ 8.0 or later queue manager, but none of the features of an IBM WebSphere MQ 7.0 or later queue manager, such as read ahead and streaming, are used.

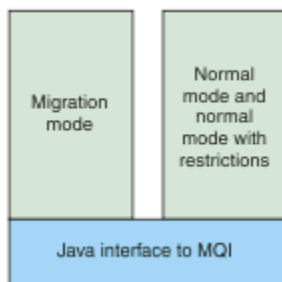


Figure 89. Messaging provider modes

Procedure

To configure the **PROVIDERVERSION** property for a specific connection factory:

- To configure the **PROVIDERVERSION** property using IBM 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, see [Creating 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 concepts

[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 IBM MQ messaging provider mode 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 JMS application uses normal mode. This mode uses all the features of an IBM MQ queue manager to implement JMS.

7 - normal mode with restrictions

The JMS application 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 and Jakarta Messaging objects using the administration tools”](#) on page 773.

Related concepts

[IBM MQ messaging provider](#)

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.

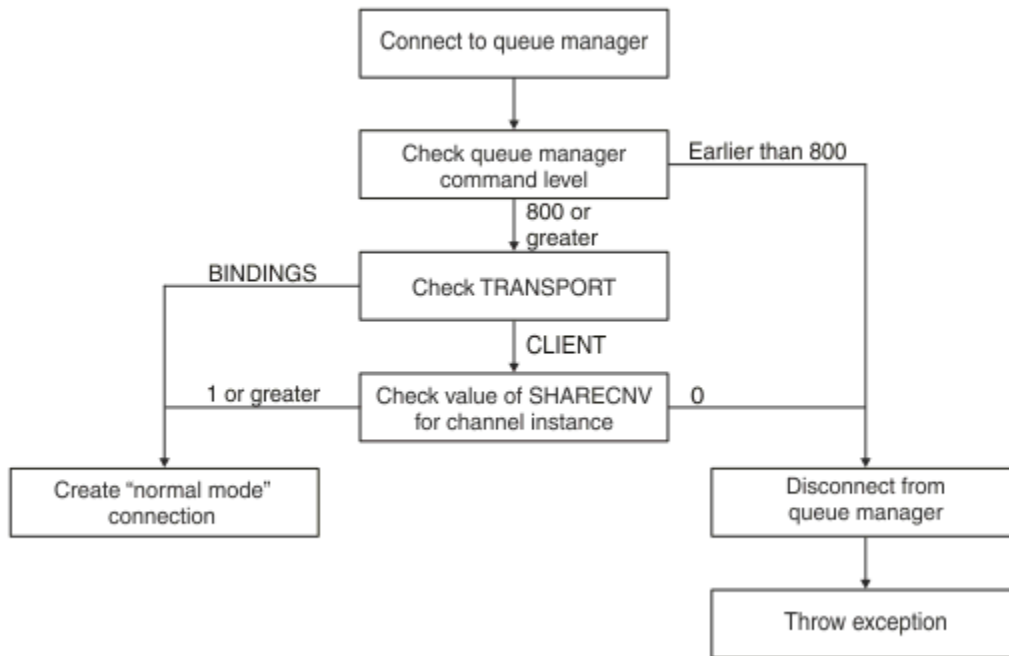


Figure 90. PROVIDERVERSION normal mode

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 JM5CC5007.

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 or later 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 .

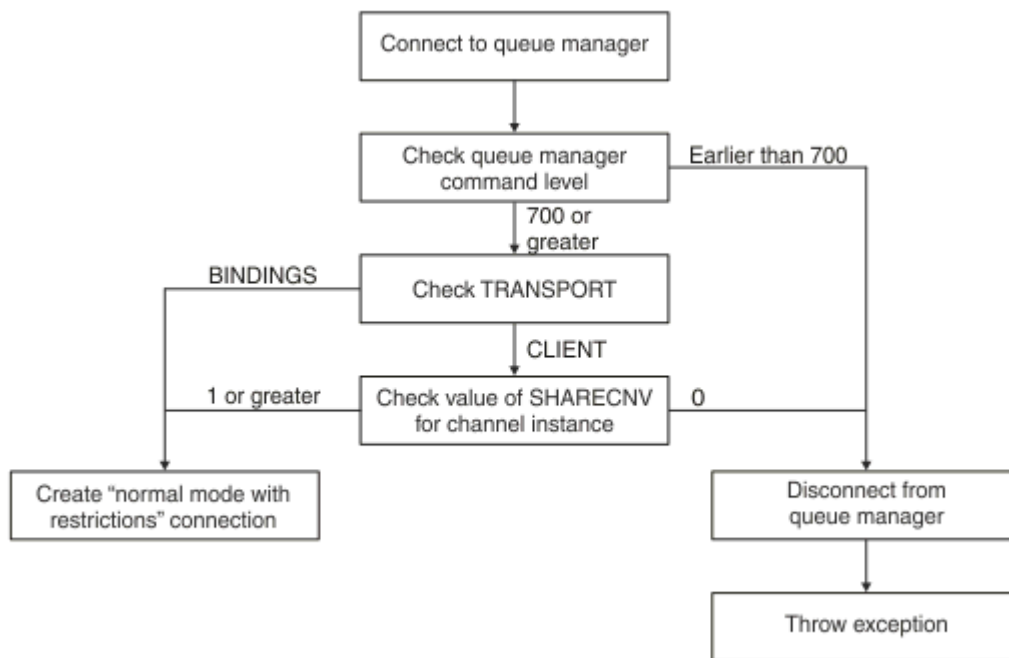


Figure 91. *PROVIDERVERSION normal mode with restrictions*

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 JMSSC5007.

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.

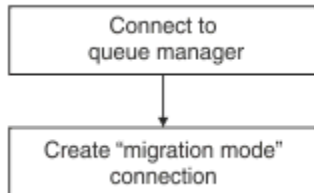


Figure 92. *PROVIDERVERSION migration mode*

If you want to connect to WebSphere Message Broker 6.0 or WebSphere Message Broker 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 or later client connecting to an IBM MQ 8.0 or later queue manager on a distributed platform,  or an IBM MQ for z/OS 8.0 or later queue manager, 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 JMSSC5007.

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.

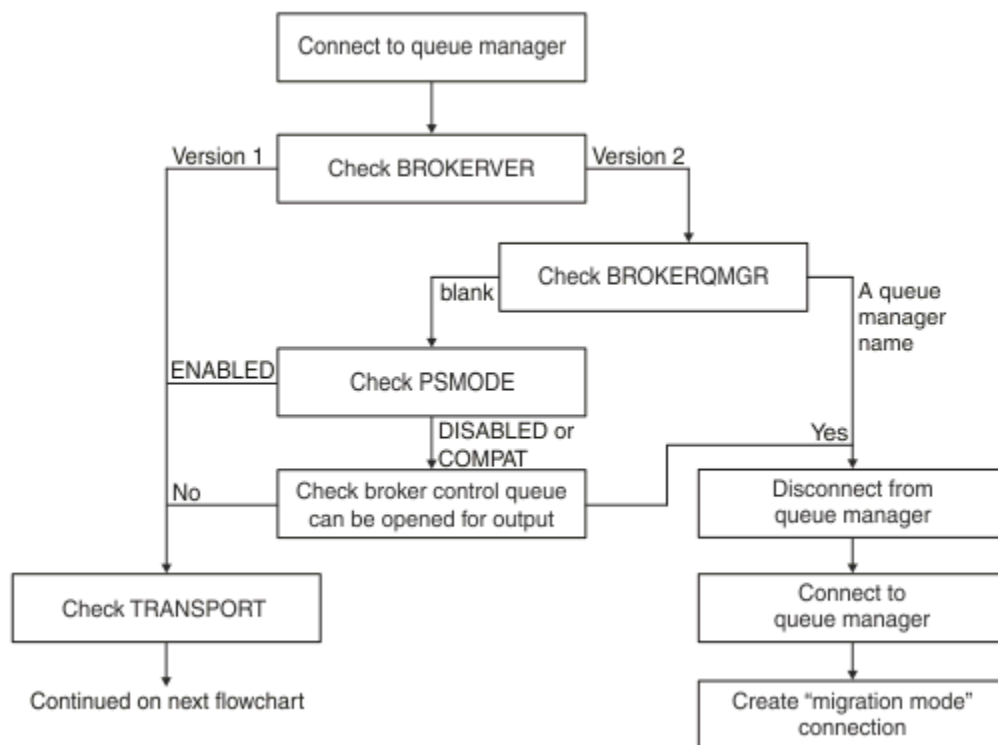


Figure 93. PROVIDERVERSION unspecified

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.

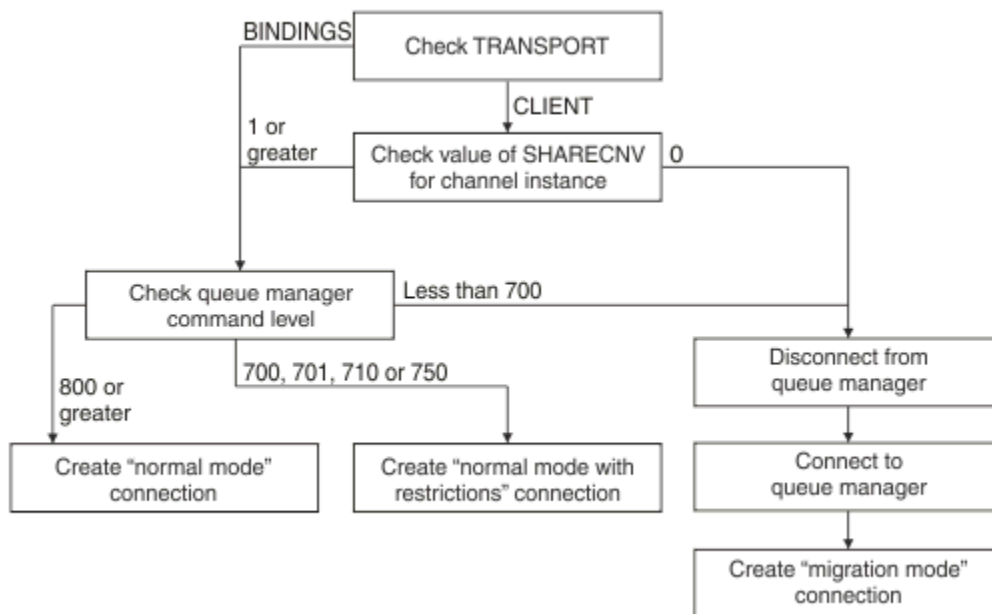


Figure 94. PROVIDERVERSION unspecified (continued)

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 750, a normal mode with restrictions connection to the queue manager is created.

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](#)

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 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 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 7.0.0

[IBM MQ messaging provider connection factory settings](#)

[**createWMQConnectionFactory** command](#)

[IBM MQ activation specification settings](#)

[**createWMQActivationSpec** command](#)

Removing WebSphere Application Server durable subscriptions

When you use the IBM MQ messaging provider with WebSphere Application Server 7.0 and WebSphere Application Server 8.0, durable subscriptions created by message-driven bean applications bound to activation specifications are not removed. Durable subscriptions can be removed using either the IBM MQ Explorer or an IBM MQ command line utility.

About this task

A message-driven bean application that removes a durable subscription can be configured to use either a listener port or an activation specification, provided that the application is running inside a WebSphere Application Server 7.0 or WebSphere Application Server 8.0 instance that uses [IBM MQ messaging provider normal mode](#) to connect to IBM MQ.

If the message-driven bean application is bound to a listener port, then the IBM MQ messaging provider creates the durable subscription for the application the first time the application is started. The durable subscription is removed when the message-driven bean application is uninstalled from an application server, and the application server restarted.

A message-driven bean application that is bound to an activation specification works 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 an IBM 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 IBM 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 IBM MQ Publish/Subscribe engine will have a Subscription Name associated with them.

Durable subscriptions created by the WebSphere Application Server IBM 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 IBM 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 activation 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 IBM 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 IBM 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 IBM 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 IBM 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 Application Server 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 IBM MQ Explorer or the IBM MQ command line utility **runmqsc**.

To delete an "orphaned" durable subscription using the IBM MQ Explorer:

1. Highlight the entry for the subscription
2. Right click the entry, and select **Delete...** from the menu. A confirmation window appears.
3. Check the subscription name displayed in the confirmation window is correct, and click **Yes**.

The IBM MQ Explorer now deletes the subscription from the Publish/Subscribe engine, and cleans 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 IBM MQ command line utility **runmqsc**, the command **delete sub** must be run:

1. Open a command prompt session
2. Navigate to the *IBM 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 Managed File Transfer

You can configure the features of Managed File Transfer after installation.

You can take advantage of IBM MQ high availability solutions to improve the resiliency of your Managed File Transfer configuration. If your agents use replicated data queue managers (RDQMs), then you must configure them to use the floating IP address feature. This means that agents use the same IP address to communicate with whichever of the three RDQM instances is currently running and automatically

reconnect on failover (see [RDQM high availability](#) and [Creating and deleting a floating IP address](#)). If you use the multi-instance queue manager solution, then applications use a different IP address to communicate with each instance, which is handled by client reconnect on failover (see [“Multi-instance queue managers”](#) on page 540 and [“Channel and client reconnection”](#) on page 594).

Related concepts

[Hints and tips for using Managed File Transfer](#)

Related tasks

[Monitoring MFT resources](#)

[Customizing MFT with user exits](#)

[Configuring MQMFTCredentials.xml](#)

[Securing Managed File Transfer](#)

[Specifying programs to run with MFT](#)

[Troubleshooting Managed File Transfer](#)

[Administering Managed File Transfer](#)

Related reference

[MFT commands](#)

[The MFTagent.properties file](#)

[MFT recovery and restart](#)

MFT configuration options on Multiplatforms

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

You can have multiple sets of configuration options, each set of configuration options contains a set of directories and properties files. The values defined in these properties files are used as the default parameters for all Managed File Transfer commands, unless you explicitly specify a different value on the command line.

To change the default set of configuration options that you are using you can use the **fteChangeDefaultConfigurationOptions** command. To change the set of configuration options that you are using for an individual command you can use the **-p** parameter with any Managed File Transfer command.

The name of a set of configuration options is the name of the coordination queue manager, and it is recommended that this is not changed. However, it is possible to change the name of a set of configuration options but you must change the name of the `config` and `logs` directories. In the following examples, the name of the set of configuration options is represented as *coordination_qmgr_name*.

Configuration options directory structure

When you configure the product, directories and properties files are created in the following structure in the configuration directory. You can also change these directories and properties files with the following commands: **fteSetupCoordination**, **fteSetupCommands**, **fteChangeDefaultConfiguration**, and **fteCreateAgent**.

```
MQ_DATA_PATH/mqft/  
  config/  
    coordination_qmgr_name/  
      coordination.properties  
      command.properties  
    agents/  
      agent_name/  
        agent.properties  
      exits  
    loggers/  
      logger_name  
        logger.properties  
  installations/
```

```
installation_name/  
installation.properties
```

The *coordination_qmgr_name* directory is a configuration options directory. There can be more than one configuration options directory in the configuration directory. The *agent_name* directory is an agent directory. In addition to containing the *agent.properties* file, this directory contains the *exits* directory, which is the default location for user exit routines and various XML files generated by the **fteCreateBridgeAgent** and **fteCreateCDAgent** commands. There can be more than one agent directory in the agents directory of a set of configuration options.

Properties files

installation.properties

The *installation.properties* file specifies the name of your default set of configuration options. This entry points Managed File Transfer to a structured set of directories and property files that contain the configuration to use. Typically the name of a set of configuration options is the name of the associated coordination queue manager. For more information about the *installation.properties* file, see [The MFT installation.properties file](#).

coordination.properties

The *coordination.properties* file specifies the connection details to the coordination queue manager. Because several Managed File Transfer installations might share the same coordination queue manager, you can use a symbolic link to a common *coordination.properties* file on a shared drive. For more information about the *coordination.properties* file, see [The MFT coordination.properties file](#).

command.properties

The MFT *command.properties* file specifies the command queue manager to connect to when you issue commands and the information that Managed File Transfer requires to contact that queue manager. For more information about the *command.properties* file, see [The MFT command.properties file](#).

agent.properties

Each Managed File Transfer Agent has its own properties file, *agent.properties*, that must contain the information that an agent uses to connect to its queue manager. The *agent.properties* file can also contain properties that alter the behavior of the agent. For more information about the *agent.properties* file, see [The MFT agent.properties file](#).

logger.properties

The *logger.properties* file specifies the configuration properties for the loggers. For more information about the *logger.properties* file, see [MFT logger configuration properties](#).

Properties files and code pages

The content of all the Managed File Transfer properties files must remain in US English because of a limitation of Java. If you edit properties files on a non-US English system, you must use Unicode escape sequences.

Related reference

[SSL/TLS properties for MFT](#)

[Java system properties for MFT](#)

[fteChangeDefaultConfigurationOptions](#)

[fteSetupCommands: create the MFT command.properties file](#)

[fteSetupCoordination](#)

[fteCreateAgent](#)

MFT configuration options on z/OS

The Managed File Transfer configuration options on z/OS are the same as the options for distributed platforms.

For more information about configuration options on [Multiplatforms](#), see “MFT configuration options on Multiplatforms” on page 806.

On z/OS, the configuration location is defined by the environment variable BFG_DATA. If a configuration does not already exist under the z/OS UNIX System Services directory that is referenced by BFG_DATA, the BFGCUSTOM JCL script of an MFT command PDSE library data set generates the jobs required to create the configuration. The configuration is then created when you run these generated jobs. Configuration creation relies on BFG_DATA referencing an existing directory that is accessible.

You can also create and maintain a configuration by using the same **fte** commands that are available on both Multiplatforms and z/OS. For a list of the **fte** commands, see [MFT commands](#).

Related concepts

“MFT configuration options on Multiplatforms” on page 806

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

“Creating an agent” on page 824

You need to copy the PDSE to make the agent-specific PDSE, for example *user.MFT.AGENT1*. Copy the PDSE from a previous agent or logger configuration, if they exist. If this is your first configuration, copy the PDSE supplied with MFT.

“Defining the coordination queue manager” on page 822

Managed File Transfer requires a queue manager to be created that acts as the coordination queue manager.

Related tasks

Configuring MQMFTCredentials.xml on z/OS

“Updating an existing MFT Agent or Logger command data set on z/OS” on page 825

You can update an Managed File Transfer command PDSE library data set that is created from the Managed File Transfer command template data set.

Downloading and configuring Redistributable Managed File Transfer components

The Redistributable Managed File Transfer package provides the Redistributable Managed File Transfer Agent, which you can configure to connect to an existing IBM MQ infrastructure and enable users to transfer files without needing to install IBM MQ. From IBM MQ 9.3.0, the redistributable package also includes the Redistributable Managed File Transfer Logger.

Before you begin

For information about redistributable license terms for the Redistributable Managed File Transfer Agent and Redistributable Managed File Transfer Logger, see [IBM MQ Redistributable Components](#).

The Redistributable Managed File Transfer package components provide the functionality of Managed File Transfer with these exceptions:

- For the Redistributable Managed File Transfer Agent, bindings mode connection to the coordination, command, and agent queue managers, is not supported, you must use client mode connection. When you are issuing commands, you must provide the parameters that are optional when you are using the Managed File Transfer that is installed as part of IBM MQ: queue manager host, port, name, and channel name.
- The Redistributable Managed File Transfer Logger supports FILE type loggers only, connecting in client mode to the coordination queue manager only. Client mode connection to the coordination queue

manager for a database logger is not supported. If you require a bindings mode connection, you must use a standard installation of IBM MQ.

- From IBM MQ 9.3.0, the **fteCreateCDAgent.cmd** command is not included. For a complete list of available commands, see [Installed MFT command sets](#).
- Managed File Transfer Connect:Direct® is not supported.
- IBM MQ Explorer is not included.

Windows You must install the Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019 libraries, available from Microsoft, on your system to use the Redistributable Managed File Transfer Agent. See [The latest supported Visual C++ downloads](#).

From IBM MQ 9.3.0, the Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019 libraries are also required for the Redistributable Managed File Transfer Logger.

Note: Advanced Message Security is not supported with the Redistributable Managed File Transfer package.

About this task

You can optionally download the Redistributable Managed File Transfer package and configure the Redistributable Managed File Transfer Agent to connect to an existing IBM MQ infrastructure to allow users to transfer files between their local environment and the existing IBM MQ infrastructure without the need for them to install IBM MQ to get the Managed File Transfer functionality.

From IBM MQ 9.3.0, the Redistributable Managed File Transfer package also includes the Redistributable Managed File Transfer Logger, which enables you to set up a file logger to connect in client mode to the coordination queue manager.

Procedure

1. Download the [IBM MQ redistributable Managed File Transfer Agent package](#) from Fix Central.

- a) Choose the package for your operating system.

The archive or .zip file names describe the file contents and equivalent maintenance levels. The file names are in the following format:

- **Windows** `V.R.M.F-IBM-MQFA-Redist-Win64`
- **Linux** `V.R.M.F-IBM-MQFA-Redist-LinuxX64`
- **Linux** `V.R.M.F-IBM-MQFA-Redist-LinuxS390X`
- **Linux** `V.R.M.F-IBM-MQFA-Redist-LinuxPPC64LE`

where *V.R.M.F* is the version number, for example 9.2.0.0 or 9.2.1.0.

- b) Identify the directory where you want to extract the package, for example:

- **Windows** `C:\MFTZ`
- **Linux** `/home/MFTZ`

2. Extract the contents of the downloaded package:

- **Windows** On Windows, use the Windows Explorer tools to extract.
- **Linux** On Linux, extract and untar as follows:

```
gunzip V.R.M.F-IBM-MQFA-Redist-LinuxX64.tar.gz
```

and then

```
tar xvf V.R.M.F-IBM-MQFA-Redist-LinuxX64.tar
```

where *V.R.M.F* is the version number, for example 9.3.0.0 or 9.3.1.0.

The following directories are created:

-   **bin**: Contains all the required MFT commands
-  **bin64**: Contains required libraries that are needed for Windows 64-bit OS support
-   **java**: Contains the IBM JRE and IBM MQ libraries
-   **licenses**: Contains the license files
-  **META-INF**: Contains files that have code signing information
-   **mqft**: Contains **ant** and **lib** directories that are required for Ant support and for core MFT function support
-   **swtag**: Contains **swidtag** file that is required by license managers to identify the installations on the machine

What to do next

You are ready to configure a Managed File Transfer Agent. For the next steps, see [“Creating the initial configuration for the Redistributable Managed File Transfer Agent”](#) on page 810.

From IBM MQ 9.3.0, you can also configure a Managed File Transfer Logger. For the next steps for configuring the logger, see [“Creating the initial configuration for the Redistributable Managed File Transfer Logger”](#) on page 812.

Related reference

[Possible errors when configuring the Redistributable Managed File Transfer components](#)

Creating the initial configuration for the Redistributable Managed File Transfer Agent

You can configure a Managed File Transfer Agent to connect to an existing IBM MQ configuration.

Before you begin

Make sure that you download and extract the contents of the Redistributable Managed File Transfer Agent package. For more information, see [“Downloading and configuring Redistributable Managed File Transfer components”](#) on page 808.

About this task

You first create the environment that the Redistributable Managed File Transfer Agent needs. You can then set up the connectivity with the queue manager that is running on the IBM MQ server, then configure an agent and the agent queue manager, before starting and verifying the agent.

From IBM MQ 9.3.0, the environment that you create is shared with the Redistributable Managed File Transfer Logger. For more information, see [“Creating the initial configuration for the Redistributable Managed File Transfer Logger”](#) on page 812.

Procedure

1. Create the environment for the Redistributable Managed File Transfer Agent.

When you run the **fteCreateEnvironment** command, the MFT data directory with the configuration information for MFT agents is created. Ensure that you are in the **bin** directory that was created

when you extracted the downloaded Redistributable Managed File Transfer Agent component. Run the following command:

• **Windows**

```
fteCreateEnvironment.cmd -d datapath location
```

• **Linux**

```
. ./fteCreateEnvironment -d datapath location
```

This command takes the following optional parameters:

-d

This parameter specifies the location for data path where the MFT configuration is created, stored, and maintained. If you run the **fteCreateEnvironment** without specifying the data location, the directory `mftdata` is created in the location where the Redistributable Managed File Transfer Agent is extracted.

Note: If the redistributable agent is going to run as a Windows service, then the **BFG_DATA** environment variable needs to be set in the system environment for the service to work.

-n installation name

This parameter is used for specifying the name of an IBM MQ installation, or a unique name.

Examples of situations in which you might want to use this parameter are:

- If you want to quickly test a new function or feature using the redistributable package with the existing configuration where agents have been configured to connect to queue manager in clients mode only. (Note that this parameter does not apply to any agent that is configured connect to a queue manager in bindings mode.)
- If you are migrating from a standard Managed File Transfer installation to a Redistributable Managed File Transfer Agent package, and you want to use the same configuration as the one that was created by the standard installation. This is the case where standard Managed File Transfer has been installed but is connecting to an agent queue manager running on another machine.

The default installation name variable is **BFG_INSTALLATION_NAME**.

For more information about the **fteCreateEnvironment** command, see [fteCreateEnvironment \(set up environment for Redistributable Managed File Transfer Agent\)](#).

You can also set the environment variable **BFG_DATA** with the data path location:

```
BFG_DATA=Datapath location
```

Before creating, starting, and stopping an agent, or any other commands, you must ensure that the **BFG_DATA** variable is set to the correct data path location.

2. Set up the IBM MQ connectivity.

a) Set up the coordination queue manager by using the **fteSetupCoordination** command.

The **fteSetupCoordination** command creates the set that is up required for coordination queue managers and the directories that are needed for further configuration. The Redistributable Managed File Transfer Agent works in client mode so you must provide extra parameters with this command to avoid an error, as bindings mode is not supported.

```
fteSetupCoordination -coordinationQMGr PRMFTDEM02  
-coordinationQMGrHost 9.121.59.233 -coordinationQMGrPort 3002  
-coordinationQMGrChannel SYSTEM.DEF.SVRCONN
```

For more details and steps for using the **fteSetupCoordination** command, see [fteSetupCoordination](#). For information on how to configure the coordination queue manager, see [“Configuring the coordination queue manager for MFT” on page 851](#).

b) Create and set up the command queue manager:

```
fteSetupCommands -p PRMFTDEM02 -connectionQMgrHost 9.121.59.233  
-connectionQMgrPort 3002 -connectionQMgrChannel SYSTEM.DEF.SVRCONN  
-connectionQMgr PRMFTDEM02 -f
```

For more details and steps for using the **fteSetupCommands** command, see [fteSetupCommands: create the MFT command.properties file](#).

3. Create an MFT Agent definition for an end point.

```
fteCreateAgent -p PRMFTDEM02 -agentQMgrHost 9.121.59.233  
-agentQMgrPort 3002 -agentQMgrChannel SYSTEM.DEF.SVRCONN  
-agentName AGENT.TRI.BANK -agentQMgr PRMFTDEM02 -f
```

For more information on using the **fteCreateAgent** command to configure an agent and the agent queue manager, see [fteCreateAgent](#).

Note: You need to use the MQSC commands that are displayed as part of the command output to define the agent objects on the agent queue manager, otherwise the instructions in Step “4” on page 812 will not work.

In steps “2” on page 811 and “3” on page 812 for each agent, you create queue and topic definitions on the agent queue manager.

4. Start the agent and you are ready to transfer files.

```
fteStartAgent -p PRMFTDEM02 AGENT.TRI.BANK
```

You can verify the status of the agent by running the following command:

```
fteListAgents
```

For more details on using the **fteListAgents** command, see [fteListAgents](#).

What to do next

If you want to configure the Redistributable Managed File Transfer Logger, complete the steps in [“Creating the initial configuration for the Redistributable Managed File Transfer Logger” on page 812](#).

Related concepts

[“Configuring Managed File Transfer” on page 805](#)

You can configure the features of Managed File Transfer after installation.

[“MFT configuration options on Multiplatforms” on page 806](#)

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Related reference

fteCreateTransfer: [start a new file transfer](#)

Creating the initial configuration for the Redistributable Managed File Transfer Logger

You can configure a FILE type Managed File Transfer Logger to connect to a coordination queue manager in client mode.

Before you begin

Make sure that you download and extract the contents of the Redistributable Managed File Transfer Agent package. From IBM MQ 9.3.0, this package also includes the Redistributable Managed File Transfer Logger. For more information, see [“Downloading and configuring Redistributable Managed File Transfer components” on page 808](#).

About this task

The Redistributable Managed File Transfer Agent and Redistributable Managed File Transfer Logger share the same environment. After this environment has been created and the IBM MQ connectivity has been set up, you can create and start the logger.

Procedure

1. Make sure that the shared environment for the Redistributable Managed File Transfer Agent and Redistributable Managed File Transfer Logger has been created as described in Step “1” on page 810 and the IBM MQ connectivity has been set up as described in Step “2” on page 811 of [“Creating the initial configuration for the Redistributable Managed File Transfer Agent”](#) on page 810.

2. Create a file logger by using the **fteCreateLogger** command.

For example:

```
fteCreateLogger FILELOGGER -loggerType FILE -loggerQMGr PRMFTDEMO2  
-loggerQMGrHost 9.121.59.233 -loggerQMGrPort 3003 -loggerQMGrChannel SYSTEM.DEF.SVRCONN  
-fileSize 20MB -fileCount 10 -fileLoggerMode CIRCULAR
```

For more information on using the **fteCreateLogger** command, see [fteCreateLogger](#).

3. Start the logger by using the **fteStartLogger** command.

For more information on the **fteStartLogger** command, see [fteStartLogger](#).

Related concepts

[“Configuring Managed File Transfer”](#) on page 805

You can configure the features of Managed File Transfer after installation.

[“MFT configuration options on Multiplatforms”](#) on page 806

Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Upgrading Redistributable Managed File Transfer components

You can upgrade the Redistributable Managed File Transfer components by downloading a new Redistributable Managed File Transfer package.

Before you begin

For information about redistributable license terms for the Redistributable Managed File Transfer Agent and Redistributable Managed File Transfer Logger see [IBM MQ Redistributable Components](#).

Note: Advanced Message Security is not supported with the Redistributable Managed File Transfer package.

About this task

If you have already installed the Redistributable Managed File Transfer components, you can upgrade them by downloading a new redistributable package and extracting the contents into the same location.

Procedure

1. Download the [IBM MQ Redistributable Managed File Transfer package](#) for your operating system from Fix Central.
2. Stop all Managed File Transfer agents and the logger. Wait for any running Managed File Transfer commands to complete.

3. Back up your existing installation by renaming the directory it has been installed into.
4. Create a new directory with the same name as the previous installation directory.
5. Extract the contents of the redistributable package into the new directory.
6. When you are happy that everything is working with the new installation, delete the directory containing the previous installation created in Step 3.



Creating an MFT Agent or Logger command data set

You can create a PDSE data set of commands from the Managed File Transfer command template data set for a specific Managed File Transfer Agent or Managed File Transfer Logger for a specific coordination.

About this task

Complete the following steps:

Procedure

1. Make a copy of the MFT command template PDSE library data set SCSQFCMD.
SCSQFCMD must be copied into a new library, for example *prefix.agent*. JCL. You can use an updated version of the SCSQFCMD(BFGCOPY) member with the following replacements:
 - Replace *++supplied-library++* with the fully qualified name of the SCSQFCMD PDSE.
 -  Replace *++service-library++* with the fully qualified name of the new MFT command PDSE library data set. The *++service-library++* is the output data set for the agent or logger service that is created.
2. For the new MFT command PDSE library data set, edit the member BFGCUSTM, which is a JCL script to customize the commands for the agent or logger. Each variable is specified in the format: *++variable name++*, which you must replace with its required value. For a description of the various JCL variables, see [“z/OS JCL variables” on page 826](#). The BFGSTDIN DD statement defines variables in three categories: Variables, Properties, and Environment. The statement has the following format:

```
[Variables]
variable1=value1
variable2=value2
....
variableN=valueN
[Properties]
property1=property value1
property2=property value2
...
propertyN=property valueN
[Environment]
custom_variable1=value1
custom_variable2=value2
....
custom_variableN=valueN
```

Variables define the set of setup and environment variables that are required for each command.

Properties define overrides for the MFT configuration properties. You can add agent and logger properties as required to customize the agent or logger for your environment. For a list of all properties, see [“Configuration properties files” on page 836](#). This facility is provided to save having to access the MFT configuration properties files, which are maintained as z/OS UNIX System Services files.

Environment defines any additionally required custom environment variables.

3. Submit job BFGCUSTM for the new MFT command PDSE library data set. This job generates the set of JCL commands, as new members of the PDSE, appropriate for the agent or logger. For a full list of the commands, see [“z/OS agent and logger command JCL scripts” on page 829](#).

Job BFGCUSTM updates the library containing the JCL which includes a DD statement with DISP=OLD. You must exit the editor after submission to allow the job to execute.

Examine the output job log to check that the JCL script ran successfully. If there are any failures, correct them and submit the BFGCUSTM job again.

The BFGCUSTM JCL script also updates the z/OS UNIX System Services MFT configuration properties files as necessary to keep the files in step. If the configuration defined by the CoordinationQMgr property does not exist, warning messages are output and you must run the generated BFGCFGR and BFGCMCR jobs to create the configuration properties files. You must run BFGAGCR for an agent, and BFGLGCRS for a logger edit. If the specified configuration already exists, the configuration is updated with any properties as defined in the BFTCUSTM JCL script.

Related concepts

[“MFT configuration options on z/OS” on page 808](#)

The Managed File Transfer configuration options on z/OS are the same as the options for distributed platforms.

Related tasks

[“Updating an existing MFT Agent or Logger command data set on z/OS” on page 825](#)

You can update an Managed File Transfer command PDSE library data set that is created from the Managed File Transfer command template data set.

Configuring Managed File Transfer for z/OS

Managed File Transfer for z/OS requires customization to enable the component to operate correctly.

About this task

You need to:

1. Edit a PDSE member to specify configuration data
2. Define the coordination queue manager.
3. Define the command queue manager
4. Configure one or more agents
5. Optionally: configure a logger task to store data in Db2

The sequence of tasks you need to perform is detailed in the following topics.

Related concepts

[“Reviewing the MFT configuration” on page 815](#)

You need to review the configuration of your system before you begin.

Related tasks

[Installing IBM MQ Advanced for z/OS](#)

Reviewing the MFT configuration

You need to review the configuration of your system before you begin.

Managed File Transfer (MFT) requires one or more queue managers to act in the following roles for each defined MFT configuration:

- A coordination queue manager, which maintains information on the status of each agent in the configuration published to a topic on the coordinator.
- One or more command or connection queue managers that act as the entry point to the IBM MQ network for MFT commands.
- One or more agent queue managers that provide the communication between an MFT agent and the IBM MQ network.

Each of the above roles can be performed by a separate queue manager, or you can combine the roles, so that, in the simplest configuration, all roles are performed by a single queue manager.

If you are adding a z/OS queue manager to an existing MFT environment you need to define connectivity between the z/OS queue manager and the other queue managers in the configuration. You can achieve this with manually defined transmission queues, or by the use of clustering.

Each MFT agent communicates with a single queue manager. If multiple agents communicate with the same queue manager, then the agent queue manager will have multiple queues defined for each agent:

- `SYSTEM.FTE.COMMAND.agent_name`
- `SYSTEM.FTE.DATA.agent_name`
- `SYSTEM.FTE.REPLY.agent_name`
- `SYSTEM.FTE.STATE.agent_name`
- `SYSTEM.FTE.EVENT.agent_name`
- `SYSTEM.FTE.AUTHAGT1.agent_name`
- `SYSTEM.FTE.AUTHTRN1.agent_name`
- `SYSTEM.FTE.AUTHOPS1.agent_name`
- `SYSTEM.FTE.AUTHSCH1.agent_name`
- `SYSTEM.FTE.AUTHMON1.agent_name`
- `SYSTEM.FTE.AUTHADM1.agent_name`

Note that you can define generic security profiles, where you use a profile such as `SYSTEM.FTE.COMMAND.*`, or you can define specific profiles for each agent.

Related concepts

[“Before you start configuring MFT for z/OS” on page 816](#)

Managed File Transfer (MFT) configuration uses files in z/OS UNIX System Services (z/OS UNIX) and PDSE data sets.

Related reference

[MFT system queues and the system topic](#)

Before you start configuring MFT for z/OS

Managed File Transfer (MFT) configuration uses files in z/OS UNIX System Services (z/OS UNIX) and PDSE data sets.

Most of the configuration and operation is done using JCL from a PDSE and you need to be familiar with working in a z/OS UNIX environment.

You can access OMVS from ISPF or you can use a Telnet type session using commands on your workstation, for example, Telnet Putty or SSH.

If you use OMVS from ISPF you can use the standard ISPF editor and browse commands **oedit** and **obrowse**.

You need to be familiar with the following z/OS UNIX commands

Table 41. Common z/OS UNIX commands	
Command	Function
<code>chmod xxx path</code>	Change files access permissions.
<code>df -k path</code>	Reports how much free space remains in the files system. -k reports the free space in KB.
<code>du -kt path</code>	Reports the sizes of directories under path. Size reported in KB.
<code>find path -name xxx</code>	Search for file named xxxx in the path directory. xxx is case sensitive and can be like *zzz.

Table 41. Common z/OS UNIX commands (continued)	
Command	Function
ls -ltrd directory	Lists information about the specified directory rather than the files in the directory.
ls -ltr path	Lists information about the files in path.
obrowse filename	Browse the file name.
oedit filename	Edit a file in OMVS.

Review the items in the following table and complete the table with the appropriate entries for your enterprise. You need these values when you edit member `BFGCUSTM`.

Table 42. Parameters needed for member BFGCUSTM		
Name	Example data	Comments
ADMIN_JOB1		Job card. All jobs are generated with the same JCL card.
armELEMENT	If ARM is being used, use the ARM ELEMENT value specified in the ARM policy for this agent or logger. If ARM is not being used, set this parameter to blank; for example, armELEMENT=	
armELEMTYPE	If ARM is being used, use the ARM ELEMTYPE specified in the ARM policy. For example, armELEMTYPE=SYSBFGAG for an agent or armELEMTYPE=SYSBFGLG for a logger. If ARM is not being used, set this parameter to blank; for example, armELEMTYPE=	
BFG_DATA		Complete as necessary
BFG_GROUP_NAME	MQM	
BFG_JAVA_HOME	/java/java80/	
BFG_JVM_PROPERTIES	V 10.0.0 From IBM MQ 10.0.0, if you are using IBM Semeru Runtime 21 or later, set the value to <code>-Dfile.encoding=IBM-1047</code> or <code>-Dfile.encoding=COMPAT</code> . This value is required because IBM Semeru Runtime 21, and later versions, use a default file encoding of UTF-8 but MFT requires a file encoding of IBM-1047.	Complete as necessary. These properties are passed to the Java virtual machine.
BFG_PROD	/usr/lpp/mqm/V9R4M0/mqft	The full path to the mqft directory under the IBM MQ for z/OS UNIX System Services Components directory.
BFG_WTO	YES	To get an MFT message on the syslog.

Table 42. Parameters needed for member BFGCUSTM (continued)		
Name	Example data	Comments
CLEAN_AGENT_PROPS	-trs	This parameter specifies the options that will be used to clean an agent when the BFGAGCL member is run. For more information on the valid values for this parameter, see fteCleanAgent: clean up an MFT Agent .
coordinationQMgr	MQPV	Mandatory configuration
CREDENTIAL_PATH		Used in migration
DB2_HLQ	DSND10	
DB_PROPS_PATH		Used in migration
FTE_CONFIG		Used in migration
JOBCARD1		This is the job card for the long running tasks, agents and loggers.
LIBRARY	SCEN.FTE.JCL	Name of the MFT PDSE. You need a copy for each agent or logger task.
MQ_HLQ	The high level qualifier for the IBM MQ data sets. For example CSQ940	
MQ_LANG	E	
MQ_PATH	/usr/lpp/mqm/V9R4M0	The full directory path to the installation of IBM MQ for z/OS UNIX System Services Components.
NAME	AGENT1	
OUTPUT_CLASS	*	
PATH	bin:/usr/bin:/usr/sbin	
productId	ADVANCEDVUE	This parameter is used to set the product type for which Managed File Transfer usage is to be recorded. For information about the valid values for this parameter, see fteSetProductId: set z/OS SCRT recording product id .
QMGR	MQPV	
SERVICE_TYPE	AGENT or LOGGER	
TMPDIR	/tmp	Read and write accessible z/OS UNIX path for temporary files.

In addition you must review the following variables and supply values where necessary:

- coordinationQMgrHost=
- coordinationQMgrPort=
- coordinationQMgrChannel=
- connectionQMgr=

- connectionQMgrHost=
- connectionQMgrPort=
- connectionQMgrChannel=

These properties are common to the AGENT or LOGGER.

Note: Host, Port, and Channel are required for client connection but should remain blank for a bindings connection on the local machine.

Related concepts

[“Items to check” on page 819](#)

Ensure that you have enough disk space, a directory for storing data, and that the requisite files exist.

[“Editing member BFGCUSTM” on page 821](#)

You must edit member BFGCUSTM, and enter the values for the parameters that your enterprise uses, before you run the job.

Items to check

Ensure that you have enough disk space, a directory for storing data, and that the requisite files exist.

Check you have enough disk space

Check that you have enough disk space available on the file system where you are going to store the configuration specific files.

If an agent trace is enabled then by default it can use 100 MB of disk space.

The configuration files themselves are small, only a few KB in size.

If you are planing on using two agents and a logger then you need at least 300 MB. You can use the command **df -k path**, where path is the location of the installation specific files. This gives the available and total space in KB.

300 MB is 307,200 KB so you should allow for at least 310,000 KB

Create and check the directory for storing Managed File Transfer data

You need a directory for storing the Managed File Transfer (MFT) data.

Check you have enough space in the file system **df -k /var**. This file system should have at least 310,000 KB available.

If you have not created this file system, use the **mkdir** command; for example **mkdir /var/mft**.

Display what permissions users have on this directory, using the command **ls -ltrd /var/mft**.

If the owner or group is not correct, use the command **chown owner:group /var/mft**.

If permissions for the group are not correct, use the following command to give the owner and the group read, write, and execute permissions. Note that the following command also gives all users read and execute permissions **chmod 775 /var/mft**.

Check the files exist and you have access to them

Use the **ls -ltr** command for the files you will be using during customization. For example:

```
ls -ltrd /java/java71_bit64_GA/J7.1_64/bin
```

gives

```
drwxr-xr-x  4 SYSTASK TSUSER   8192 Nov 15 2013 /java/java71_bit64_GA/J7.1_64/bin
```

where the **drwxr-xr-x** means

d

This is a directory.

rwX

The owner *SYSTASK* has read, write and execute access to the directory.

r-X

People in the group *TSOUSER* can read and execute files in the directory.

r-X

Universal access, that is, anyone can read or execute files in the directory.

Check the files specified in:

Table 43. Access required by users to specific files	
Path	Access required by users doing the configuration
BFG_JAVA_HOME	Read and execute
/tmp	Read and write
BFG_PROD	Read
BFG_DATA	Write
MQ_PATH	Read

Related concepts

[“Before you start configuring MFT for z/OS” on page 816](#)

Managed File Transfer (MFT) configuration uses files in z/OS UNIX System Services (z/OS UNIX) and PDSE data sets.

[“Common MFT for z/OS configurations” on page 820](#)

An overview of the different Managed File Transfer configurations

Common MFT for z/OS configurations

An overview of the different Managed File Transfer configurations

Managed File Transfer uses agents attached to a queue manager for transferring data.

MFT can use multiple queue managers:

- One or more queue managers to transfer the data.
- A commands queue manager that issues requests. For example, a request to start a transfer is sent to this queue manager, and the associated commands are routed to the MFT agents.
- A coordination queue manager that manages the work.

There are three common Managed File Transfer (MFT) configurations:

1. A single queue manager with one or more agents using local connections. This might be used to put the contents of a data set into IBM MQ queues.
2. A single queue manager with an MFT client on a distributed machine using client bindings.
3. Two queue managers connected by channels, and one or more agents on each machine. These agents can be client or local bindings.

Note the following points:

1. MFT is written in Java, with some shell scripts and JCL to configure and operate MFT.
2. Db2 status and activity can be logged, and this can be stored in Db2 tables.
3. The person configuring MFT must be familiar with z/OS UNIX System Services (z/OS UNIX). For example:
 - The directory structure with files with names like `/u/userID/myfile.txt2`

- z/OS UNIX commands, such as:

cd (change directory)

ls (list)

chmod (change file permissions)

chown (change file ownership or groups that can access the file or directory)

4. The following products are required in z/OS UNIX to be able to configure and run MFT:

- Java; for example, /java/java71_bit64_GA/J7.1_64/
- IBM MQ V920, for example /mqm/V9R2M0.
- Db2 JDBC libraries, if you want to use Db2 for status and history; for example /db2/db2v12/jdbc/lib

You need a coordination queue manager. However, you can use the same queue manager to run agents, to process commands, and for coordination. If you are using multiple queue managers, you must pick one to act as the coordinator.

Check your IBM MQ connectivity

If you have an existing MFT coordinator queue manager, you need connectivity between the queue manager where you are doing the configuration, and the coordinating and command queue managers.

Copy SCSQFCMD to create a JCL library

You need to create a JCL library for each agent and logger. The JCL contains the configuration and jobs used to create and run the agent or logger.

For each agent and logger create a copy of the IBM supplied SCSQFCMD library by editing and running the BFGCOPY member.

This library is used to define the configuration for the agent or logger and, after customization, contains jobs that can be used to create the required Managed File Transfer configuration and agent or logger.

You create member BFGCUSTM as part of this process.

Note: If you are familiar with z/OS UNIX commands, you can configure z/OS with the same commands that you use on other platforms.

Related concepts

[“Common MFT for z/OS configurations” on page 820](#)

An overview of the different Managed File Transfer configurations

[“Editing member BFGCUSTM” on page 821](#)

You must edit member BFGCUSTM, and enter the values for the parameters that your enterprise uses, before you run the job.

Editing member BFGCUSTM

You must edit member BFGCUSTM, and enter the values for the parameters that your enterprise uses, before you run the job.

See [Parameters needed for member BFGCUSTM](#), for a list of the parameters requiring specific values.

In addition you must review the following variables and supply values where necessary:

- coordinationQMGrHost=
- coordinationQMGrPort=
- coordinationQMGrChannel=
- connectionQMGr=
- connectionQMGrHost=
- connectionQMGrPort=

- connectionQMgrChannel=

These properties are common to the AGENT or LOGGER.

Note: Host, Port, and Channel are required for client connection but should remain blank for a bindings connection on the local machine.

If this is the first queue manager in your Managed File Transfer environment, and you want to use the same queue manager for coordination, commands, and running agents, set the values to the local queue manager name.

```
coordinationQMgr=MQPV
connectionQMgr=MQPV
```

where MQPV is your local queue manager name.

Submit the job, which updates the PDSE, and creates a directory structure under the specified path.

Note that this job requires exclusive use, so you need to stop using the PSDE while the job runs.

Tip: Whenever you submit job BFGCUSTM, the job replaces all the JCL files. You should rename each member you change.

Related concepts

[“Before you start configuring MFT for z/OS” on page 816](#)

Managed File Transfer (MFT) configuration uses files in z/OS UNIX System Services (z/OS UNIX) and PDSE data sets.

[“Creating an agent” on page 824](#)

You need to copy the PDSE to make the agent-specific PDSE, for example *user.MFT.AGENT1*. Copy the PDSE from a previous agent or logger configuration, if they exist. If this is your first configuration, copy the PDSE supplied with MFT.

Defining the coordination queue manager

Managed File Transfer requires a queue manager to be created that acts as the coordination queue manager.

Depending on the configuration that you have chosen, this queue manager is on the local MVS system, or on another machine. In the former case, the connections to it are bindings connections and in the latter case, they are client connections.

After you have run the configuration step successfully there are configured members in the PDSE.

Member BFGCFR defines the coordination queue manager, and this job:

1. Creates a directory structure in the Managed File Transfer (MFT) directory, and creates configuration files.
2. Runs CSQUTIL to define IBM MQ resources.

If the coordination queue manager is on a remote machine then this job step fails.

Member BCFCFCR creates files in z/OS UNIX System Services and creates MQ definitions. This job:

1. Creates an MFT topic,
2. Creates an MFT queue
3. Alters *NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)* to be *NAMES(SYSTEM.BROKER.DEFAULT.STREAM, SYSTEM.BROKER.ADMIN.STREAM, SYSTEM.FTE)*
4. Performs *ALTER QMGR PSMODE(ENABLED)*

A *DISPLAY NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)* command is issued before doing the alter.

If your NAMLIST is not the default, you should alter your name list to add SYSTEM.FTE to your namelist

Rename member BCFCFCR with your own prefix, for example, CCPCFCR, because re customizing this file replaces it.

Edit this renamed member by inserting the name of your credentials file. For example:

```
%BFGCMD CMD=fteSetupCoordination +  
-credentialsFile //'<MFTCredentialsDataSet(MemberName)>'
```

Save and submit the job. Note that if you need to resubmit the job, you need to add the *-f* option.

When this job runs it lists the IBM MQ resources it creates. You need to protect these resources.

```
DEFINE TOPIC('SYSTEM.FTE') TOPICSTR('SYSTEM.FTE') REPLACE  
ALTER TOPIC('SYSTEM.FTE') NPMGDLV(ALLAVAIL) PMSGDLV(ALLAVAIL)  
DEFINE QLOCAL(SYSTEM.FTE) LIKE(SYSTEM.BROKER.DEFAULT.STREAM) REPLACE  
ALTER QLOCAL(SYSTEM.FTE) DESCR('Stream for MFT Pub/Sub interface')  
* Altering namelist: SYSTEM.QPUBSUB.QUEUE.NAMELIST  
* Value prior to alteration:  
DISPLAY NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)  
ALTER NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST) +  
NAMES(SYSTEM.BROKER.DEFAULT.STREAM+  
,SYSTEM.BROKER.ADMIN.STREAM,SYSTEM.FTE)  
* Altering PSMODE. Value prior to alteration:  
DISPLAY QMGR PSMODE  
ALTER QMGR PSMODE(ENABLED)
```

Related tasks

[“Defining the command queue manager” on page 823](#)

You can either use the same queue manager as the coordination and command queue managers, or create a new command queue manager.

Defining the command queue manager

You can either use the same queue manager as the coordination and command queue managers, or create a new command queue manager.

About this task

You must have a command queue manager, however, you can use the same queue manager for the coordination and command queue managers. Otherwise, you need to create a new command queue manager. This can be on the same machine as the coordination queue manager, but does not have to be.

Procedure

1. Rename member BFGCMCR with your own prefix, for example, CCPCMCR.
You must rename BFGCMCR because re-customizing this file replaces it.
2. Edit the renamed member by inserting the name of your credentials file.
For example:

```
%BFGCMD CMD=fteSetupCommands +  
-credentialsFile //'<MFTCredentialsDataSet(MemberName)>' +
```

3. Save and submit the job.
Note that if you need to resubmit the job, you need to add the *-f* option.
This queue manager is used for commands such as **ftePingAgent**.
4. Review this member, submit it, and review the output.

What to do next

See [“Creating an agent” on page 824](#) for information on how you create an agent.

Related concepts

[“Defining the coordination queue manager” on page 822](#)

Managed File Transfer requires a queue manager to be created that acts as the coordination queue manager.

Related tasks

[Configuring MQMFTCredentials.xml](#)

Related reference

[MFT credentials file format](#)

Creating an agent

You need to copy the PDSE to make the agent-specific PDSE, for example *user.MFT.AGENT1*. Copy the PDSE from a previous agent or logger configuration, if they exist. If this is your first configuration, copy the PDSE supplied with MFT.

Review member BFGCUSTM and if you need to use a different credentials file, create one.

Much of the content remains the same from the customization detailed in [“Editing member BFGCUSTM” on page 821](#).

You need to change:

- //SYSEXEC DD DSN=SCEN.FTE.JCL.AGENT1
- LIBRARY to match the agent PDSE
- SERVICE_TYPE=AGENT
- NAME to be the name of the agent (matching the PDSE) JOBCARD
- Change BFG_JVM_PROPERTIES="-Xmx1024M"

Submit this job, remembering that the job requires exclusive access to the data set.

The jobs for the agent all have names of the form *BFGAG**

Rename member *BFGAGCR*. This job updates files in the Managed File Transfer directory and uses CSQUTIL to create agent specific queues in the local queue manager. Specify the name of your credentials file, for example, -credentialsFile // ' SCEN.FTE.JCL.VB(CREDOLD). If you do not specify the name, the job to start the agent does not use a credentials file.

Check the output to ensure that the process has run successfully.

Tip: Copy the path name of the *agent.properties* file from the output of the job to a member in the PDSE for the agent.

For example, copy /u/userid/fte/wmqmft/mqft/config/MQPA/agents/AGENT1/agent.properties into member AGENT.

This is useful if you need to display the properties file, and add the line /u/userid/fte/wmqmft/mqft/logs/MQPA/agents/AGENT1/logs.

This is where trace files are stored.

Related concepts

[“Defining the coordination queue manager” on page 822](#)

Managed File Transfer requires a queue manager to be created that acts as the coordination queue manager.

[“Using the agent” on page 825](#)

How you use various commands to ensure that the agent is working correctly.

Related tasks

[“Defining the command queue manager” on page 823](#)

You can either use the same queue manager as the coordination and command queue managers, or create a new command queue manager.

Using the agent

How you use various commands to ensure that the agent is working correctly.

Start the agent

Rename member BFGAGST, review the member, and submit the job.

If this works you receive message BFGAG0059I: The agent has been successfully started.

Display the active agent(s)

Rename member BFGAGLI, review the member and submit the job which uses the coordinating queue manager.

You must resolve any connectivity problems

Ping the agent to check it is working

Rename member BFGAGPI, review the member and submit the job which uses the command queue manager.

You must resolve any connectivity problems

Carry out a test transfer

See [“Performing a verification transfer” on page 831](#) for further information.

Stop the agent

Rename member BFGAGSP, review the member and submit the job.

Restart the agent using the member BFGAGST.

Related concepts

[“Creating an agent” on page 824](#)

You need to copy the PDSE to make the agent-specific PDSE, for example *user.MFT.AGENT1*. Copy the PDSE from a previous agent or logger configuration, if they exist. If this is your first configuration, copy the PDSE supplied with MFT.

Updating an existing MFT Agent or Logger command data set on z/OS

You can update an Managed File Transfer command PDSE library data set that is created from the Managed File Transfer command template data set.

Procedure

1. Edit the BFGCUSTM JCL script member and update variables and properties in the BFGSTDIN DD statement.

If you want to remove a property that was previously defined, set its value to blank, instead of removing the entry. When the BFGCUSTM JCL script is run, the specified properties are applied as an update to the actual agent and logger z/OS UNIX System Services properties files; setting a property to a blank value indicates that the property is to be removed

2. Submit job BFGCUSTM. This job generates the set of JCL commands again, appropriate for the agent or logger. For a full list of the commands, see [“z/OS agent and logger command JCL scripts” on page](#)

829. Examine the output job log to check that the JCL script ran successfully. If there are any failures, correct them and submit the BFGCUSTM job again.

Results

You can modify the generated JCL scripts and add your own logic. However, be careful when you run BFGCUSTM again because you might overwrite the custom logic.

Related concepts

“MFT configuration options on z/OS” on page 808

The Managed File Transfer configuration options on z/OS are the same as the options for distributed platforms.

Related tasks

“Creating an MFT Agent or Logger command data set” on page 814

You can create a PDSE data set of commands from the Managed File Transfer command template data set for a specific Managed File Transfer Agent or Managed File Transfer Logger for a specific coordination.

► z/OS z/OS JCL variables

You can use substitution values, JCL variables, and configuration properties in the BFGCUSTM script.

The following table lists the substitution values for the BFGCUSTM JCL script in an MFT command PDSE library data set. You must replace these substitution values with suitable values before you submit the BFGCUSTM job.

Table 44. Substitution values for the BFGCUSTM JCL script	
Substitution variable	Value
++library++	The data set name of the containing MFT command PDSE library.
++bfg_java_home++	The location of your Java installation.
++mq_path++	The path to the IBM MQ for z/OS UNIX System Services Components directory. For example, /mqm/V9R2M0. This is used to give the full path to the MFT installation, for example, /mqm/V9R2M0/mqft.

The following table describes the environment variables for the BFGSTDIN DD statement for the BFGCUSTM JCL script, in an MFT command PDSE library data set (in the [Variables] section). You must replace all variables that are specified with substitution values (that is, values enclosed in two plus signs, ++) with suitable values before you submit the BFGCUSTM job.

Table 45. Environment variables	
Environment variable	Value
LIBRARY	The data set name of the containing MFT command PDSE library.
TMPDIR	z/OS UNIX System Services directory for temporary files.
BFG_PROD	The full path to the mqft directory under the IBM MQ for z/OS UNIX System Services Components directory; For example: /mqm/V9R2M0/mqft.
BFG_DATA	The location of the data directory for Managed File Transfer for z/OS, which is the path to DATA_DIR.
BFG_JAVA_HOME	The location of your Java installation.

Table 45. Environment variables (continued)

Environment variable	Value
BFG_JVM_PROPERTIES	Sets a value for the BFG_JVM_PROPERTIES environment variable. These properties are passed to the Java virtual machine. Version From IBM MQ 10.0.0, set the value to <code>-Dfile.encoding=IBM-1047</code> or <code>-Dfile.encoding=COMPAT</code> if you are using IBM Semeru Runtime 21 or later. This value is required because IBM Semeru Runtime 21, and later versions, use a default file encoding of UTF-8 but MFT requires a file encoding of IBM-1047. In earlier IBM MQ releases, this environment variable is optional.
BFG_GROUP_NAME	The mqm file group is typically associated with MFT configuration data files and commands. Consequently, all users who are members in the mqm group can access and make changes to the MFT configuration. For more information, see File system permissions for MFT in IBM MQ. For a z/OS system, a file group is a z/OS UNIX System Services (z/OS UNIX) filesystem entity, and the mqm file group is not necessarily defined. You can associate a z/OS UNIX filesystem group for MFT configuration data files by using the BFG_GROUP_NAME environment variable. For example, at the z/OS UNIX shell prompt use: <pre>export BFG_GROUP_NAME=FTEGB</pre> which defines a group <i>FTEGB</i> to be associated with any subsequently created configuration files for the current z/OS UNIX session. You can set BFG_GROUP_NAME to a blank value, or remove it. Note: When running BFGCUSTM for the first time, if the MFT configuration is to be used by multiple user IDs, it is important that BFG_GROUP_NAME is set to a group accessible to all required user ID's. If BFGCUSTM is run again, then BFG_GROUP_NAME must not be changed (otherwise, the z/OS UNIX group file permissions for all files and directories in the directory referenced by BFG_DATA must also be changed to reflect the new BFG_GROUP_NAME setting).
BFG_WTO	z/OS logging is enabled when BFG_WTO is set to YES, ON, or TRUE. This controls whether messages that are written to the agent event log are also written to the z/OS operator log facility, which allows easier access for automation products when you run an agent from JCL. The routing code is Programmer Information (11) and the descriptor code is Informational (12).

Table 45. Environment variables (continued)

Environment variable	Value
SERVICE_TYPE	Specifies whether the MFT command library is for an agent or logger. The valid values are AGENT or LOGGER.
NAME	The name of the agent or logger for the SERVICE_TYPE value.
QMGR	The name of the local queue manager that is associated with the agent or logger for the SERVICE_TYPE value.
OUTPUT_CLASS	The output class for SYSOUT data sets. Defaults to * which requests the same output class as the MSGCLASS parameter from the job statement.
MQ_PATH	The path to the IBM MQ for z/OS UNIX Components directory.
MQ_HLQ	The high-level qualifier for IBM MQ data sets.
MQ_LANG	The language that is required.
DB2_HLQ	Optional. High-level qualifier for Db2 data sets.
JOBCARD1	Header line 1 for a JCL command job.
JOBCARD2	Header line 2 for a JCL command job.
JOBCARD3	Header line 3 for a JCL command job.
ADMIN_JOB1	Header line 1 for an admin job.
ADMIN_JOB2	Header line 2 for an admin job.
ADMIN_JOB3	Header line 3 for an admin job.
FTE_CONFIG	Existing MFT configuration for migration. Set to a blank value if migration is not required.
CREDENTIAL_PATH	Path to credentials file for migration, for example /u/user1/agent3. Required for migration commands BFGAGMG and BFGLGMG JCL scripts only. Set to a blank value if migration is not required. Note also that
DB_PROPS_PATH	Specifies the database logger properties file for migration. This option is required only if the properties file does not use the following default name and path: config_directory/coordination_qmgr/databaselogger.properties. Set to a blank value if migration is not required.

The following table describes the mandatory MFT configuration properties for the BFGSTDIN DD statement for the BFGCUSTM JCL script in an MFT command PDSE library data set. You must replace properties specified with substitution values (that is, values enclosed in two plus signs, ++) with a suitable non-blank value before you submit the BFGCUSTM job. These properties define overrides for the MFT configuration properties. You can add agent and logger properties to customize agents or loggers for your environment. For a list of all properties, see [“Configuration properties files”](#) on page 836.

Table 46. Mandatory configuration properties for the BFGSTDIN DD statement

Property	Value
coordinationQMgr	The name of the coordination queue manager for the configuration that the agent or logger is associated with.
coordinationQMgrHost	Optional. Host name of the system that the coordination queue manager is running on. If you leave the value for this property blank, a bindings mode connection is assumed.
coordinationQMgrPort	Optional. Port number that the coordination queue manager is listening on. This parameter is used only if you also specify a non-blank value for the coordinationQMgrHost property.
coordinationQMgrChannel	Optional. Channel to use to connect to the coordination queue manager. This parameter is used only if you also specify a non-blank value for the coordinationQMgrHost property.
connectionQMgr	The name of the command queue manager for the configuration that the agent or logger is associated with.
connectionQMgrHost	Optional. Host name of the system that the command queue manager is running on. If you leave the value for this property blank, a bindings mode connection is assumed.
connectionQMgrPort	Optional. Port number that the command queue manager is listening on. This parameter is used only if you also specify a non-blank value for the connectionQMgrHost property.
connectionQMgrChannel	Optional. Channel to use to connect to the command queue manager. This parameter is used only if you also specify a non-blank value for the connectionQMgrHost property.

z/OS agent and logger command JCL scripts

The set of JCL commands available in an MFT command PDSE library data set.

Table 47. JCL commands available in an MFT command PDSE library data set

Member	Description or fte command line command
BFGCOPY	Job to create a copy of this library
BFGCUSTM	Job to customize this library for agent or logger
BFGZCFGR	fteSetupCoordination
BFGZCMCR	fteSetupCommands : create the MFT command.properties file
BFGZAGCR	fteCreateAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGLGCRS	fteCreateLogger . Created only when you set the SERVICE_TYPE variable to LOGGER.

Table 47. JCL commands available in an MFT command PDSE library data set (continued)

Member	Description or fte command line command
BFGZAGST	fteStartAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGAGSTP	fteStartAgent procedure. Created only when you set the SERVICE_TYPE variable to AGENT.
BFGZAGPI	ftePingAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGZAGSP	fteStopAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGZLGST	fteStartLogger . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGLGSTP	fteStartLogger procedure. Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGZLGSP	fteStopLogger . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGZAGSH	fteShowAgentDetails . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGZLGSH	fteShowLoggerDetails . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGZCFDF	fteChangeDefaultConfigurationOptions
BFGZAGCL	fteCleanAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGZAGDE	fteDeleteAgent . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGZLGDE	fteDeleteLogger . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGZPRSH	fteDisplayVersion
BFGZAGLI	fteListAgents . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGZMNLI	fteListMonitors
BFGZSTLI	fteListScheduledTransfers
BFGZTMLI	fteListTemplates
BFGXCROB	fteObfuscate sample
BFGZRAS	fteRAS
BFGZAGTC	fteSetAgentTraceLevel . Created only when you set the SERVICE_TYPE variable to AGENT.
BFGZLGTC	fteSetLoggerTraceLevel . Created only when you set the SERVICE_TYPE variable to LOGGER.
BFGXPRAN	fteAnt sample
BFGXTRCA	fteCancelTransfer sample
BFGXMNCR	fteCreateMonitor sample

Table 47. JCL commands available in an MFT command PDSE library data set (continued)

Member	Description or fte command line command
BFGXTMCR	fteCreateTemplate sample
BFGXTRCR	fteCreateTransfer sample
BFGXMNDE	fteDeleteMonitor sample
BFGXSTDE	fteDeleteScheduledTransfer sample
BFGXTMDE	fteDeleteTemplate sample

Notes:

- The JCL, for commands that create MQSC or reference delete scripts, asks you to run a script, but the script has already been run by the job.
- BFGZRAS creates the BFGRAS member when the BGCUSTOM job is run.

Performing a verification transfer

How you carry out a transfer to check that the product is working correctly.

Rename and edit member BFGTRCRS.

1. Add a /* before the %BFGCMD CMD=fteCreateTransfer -h
2. Remove the other comments in the member.
3. Specify the current agent name for -sa and -da
4. Save the JCL
5. Submit the JCL

This JCL connects to the command queue manager.

Configuring a logging task

The logging task needs run on the same image as the coordination queue manager. You can log to Db2.

Creating a logging task

Copy the PDSE to make the logger specific PDSE. For example, user.MFT.LOGGER.

If you need to use a different credentials file, create one. See [Configuring MQMFTCredentials.xml on z/OS](#).

Review member [BFGCUSTOM](#). Note that much of the content remains the same from the previous customization.

However, you need to:

- Change //SYSEXEC DD DSN=SCEN.FTE.JCL....
- Change LIBRARY to match the agent PDSE
- Change QMGR to the name of the coordination queue manager
- Make SERVICE_TYPE=LOGGER
- Change NAME to be the name of the logger (matching the PDSE)
- Review JOBCARD and change the jobname so that the name is different from the job names of the agents.
- Review BFG_JVM_PROPERTIES="-Xmx1024M"

If you are using the Db2 logger it is useful to create a file, so that you can capture Db2 traces to help identify Db2 problems.

The name of the file is specified in the JVM properties, where the JDBC trace properties file has content such as

```
db2.jcc.traceDirectory=/u/johndoe/fte
db2.jcc.traceFile=jccTrace1
db2.jcc.traceFileAppend=false
# turn on all traces
# db2.jcc.traceLevel=-1
# turn off all traces
db2.jcc.traceLevel=0
```

Set two JVM properties

```
BFG_JVM_PROPERTIES=-Ddb2.jcc.propertiesFile=/u/.../sql.properties
-Ddb2.jcc.ssid=DBCA
```

Where `/u/.../sql.properties` is the name of your Db2 trace properties file, and `DBCA` is the name of your Db2 subsystem.

Submit this job, noting that the job requires exclusive access to the data set. The jobs for the agent all have names like `BFGLG*`.

Logging to files

For more information on logging to Db2, see [“Creating a logging task, when logging to Db2” on page 833](#)

Rename member `BFGLGCRS`. This job updates files in the Managed File Transfer (MFT) directory and uses `CSQUTIL` to create agent specific queues in the local queue manager.

The original file has command `%BFGCMD CMD=fteCreateLogger -h` which lists the syntax of the command.

To create the logger task comment out the `%BFGCMD CMD=fteCreateLogger -h` by putting `/*` in front of the statement, making sure that column one is blank.

Remove the comments from the second command and configure the statements. For example:

```
%BFGCMD CMD=fteCreateLogger  +
-p MQPH      +
-loggerQMgr MQPH      +
-loggerType FILE      +
-fileLoggerMode circular      +
-fileSize 5MB +
-fileCount 5 +
-p MQPH +
-credentialsFile //'<MFTCredentialsDataSet(MemberName)>'
LOGGER
```

Check the output to see that it has processed successfully.

Tip: Copy the pathname of the `logger.properties` file from the output of the job to a member in the PDSE of the agent.

For example copy into member `APATH`

```
/u/user_ID/fte/wmqmft/mqft/config/MQPH/loggers/LOGGER/logger.properties
```

This is useful if you need to display properties file.

Add the directory to this file:

```
/u/user_ID/fte/wmqmft/mqft/logs/MQPH/loggers/LOGGER/
```

If you are logging to file, the log files are stored in this directory, for example `LOGGER0-20140522123654897.log`.

Trace files are in the log subdirectory, for example

```
/u/user_ID/fte/wmqmft/mqft/logs/MQPH/loggers/LOGGER/logs
```

You can now [start the logging task](#).

Creating a logging task, when logging to Db2

Rename member BFGLGCRS.

This job updates files in the MFT directory and uses CSQUTIL to create agent specific queues in the local queue manager.

You need to know:

Table 48. Db2 variables	
Db2 name	Example
-dbName <i>databaseName</i>	You can get this from the location value in message DSNL004I for your Db2 subsystem
-dbDriver <i>filePath</i>	For example /db2/db2v10/jdbc/classes/db2jcc.jar
-dbLib <i>filePath</i>	For example /db2/db2v10/jdbc/lib/libdb2jcct2zos_64.so

Edit the file. The original file has command %BFGCMD CMD=fteCreateLogger -h which lists the syntax of the command.

Remove the comments from the second command and configure the statements. For example

```
%BFGCMD CMD=fteCreateLogger  +
-p MQPH  +
-loggerQMgr MQPH  +
-loggerType DATABASE  +
-dbType DB2  +
-databaseName DSNDBCP  +
-dbDriver /db2/db2v10/jdbc/classes/db2jcc.jar  +
-dbLib /db2/db2v10/jdbc/lib/  +
-credentialsFile //'<MFTCredentialsDataSet(MemberName)>' +
LOGGER
```

To create the logger task comment out the %BFGCMD CMD=fteCreateLogger -h by putting /* in front of the statement, making sure that column one is blank.

Submit the job and check the output to see that it has processed successfully.

Tip: Copy the pathname of the logger.properties file from the output of the job to a member in the PDSE of the agents.

For example copy into member APATH:

```
/u/user_ID/fte/wmqmft/mqft/config/MQPH/loggers/LOGGER/logger.properties into member USS
```

This is useful if you need to display the properties file

Trace files are in the log subdirectory, for example:

```
/u/user_ID/fte/wmqmft/mqft/logs/MQPH/loggers/LOGGER/logs
```

Creating Db2 tables

You need to create the Db2 tables. The definitions are in the z/OS UNIX System Services file mqft/sql/ftelog_tables_zos.sql.

Create a member Db2 in your PDSE. Edit this member and use the COPY command on the command line. Copy from the z/OS UNIX System Services definitions file.

Because site-specific requirements can vary greatly, this file only specifies the basic structures of the tables, and a table space where they will be located.

The table space is specified, by the SQL script, to ensure that it is created using a buffer pool with a page size sufficient to hold the largest tables rows possible. Note that attributes such as LOB locations, and so on, are not specified.

Your database administrator might want to modify a copy of this file, to define these performance-related attributes.

This file also assumes a default schema name of FTELOG, default tablespace name of FTELOGTS, and database name of FTELOGDB. You can change these names if you need, to match an existing database and any local naming conventions, by following the process described in the comments at the beginning of the file.

Important: Use online facilities like **SPUFI** to run the commands, because there are comments in the file, and batch programs like **DSNTINAD** do not accept comments.

See [Executing SQL by using SPUFI](#) for more information. In addition, CSQ45STB in SCSQPROC has sample JCL that you can customize to do the Db2 SELECT commands.

Starting the logger task

Rename, review and submit the member BFGLGST You should get the message BFGDB0023I: The logger has completed startup activities and is now running.

Logger operations

To display the logger status, Rename, review, and submit member BFGLGSH

To stop the logger, Rename, review, and submit member BFGLGSP.

Environment variables for MFT on z/OS

If you are running commands direct from the z/OS UNIX System Services (z/OS UNIX) environment, or your own JCL scripts, after customization and configuration you must set a number of environment variables before running the configuration and administration scripts provided by Managed File Transfer. You must set these variables for each user and in each environment that the scripts will be invoked from.

To avoid conflicts with other products, you can choose to create a `.wmqfterc` script in your home directory. The `.wmqfterc` script is then invoked by each of the Managed File Transfer scripts and you can use this script to provide custom environment settings for Managed File Transfer.

There is also one optional environment variable, `BFG_WTO`, that you can set to send messages to the operator log when running agents from JCL.

Table 49. Required z/OS environment variables	
Environment variable	Value
BFG_JAVA_HOME	The location of your Java installation. For more information about the levels of Java supported, see System Requirements for IBM MQ .
BFG_DATA	The location of the data directory for Managed File Transfer for z/OS. This is the path to <code>DATA_DIR</code> .

Table 49. Required z/OS environment variables (continued)	
Environment variable	Value
STEPLIB	Must include the following IBM MQ data sets: • SCSQAUTH • SCSQANLE • SCSQLOAD If you want to run the database logger component on a z/OS system, STEPLIB must also include the following Db2 data sets in the order shown: • SDSNEXIT • SDSNLOD2 • SDSNLOAD

The following is an example .profile that correctly configures the environment variables for Managed File Transfer:

```
STEPLIB=MQM.V920.SCSQAUTH:MQM.V920.SCSQANLE:MQM.V920.SCSQLOAD
PATH=/u/fteuser/bin:/u/fteuser/J7.0/bin:/bin:/usr/bin:/u/fteuser/extras/bin:/bin:$PATH
BFG_JAVA_HOME=/u/fteuser/J7.0
BFG_DATA=/u/fteuser/DATA_DIR
export PATH STEPLIB BFG_JAVA_HOME BFG_DATA
```



Attention: The LIBPATH environment variable is no longer needed when calling **fte*** commands from a z/OS UNIX environment, and should be removed from any existing .wmqfterc script

Optionally, you can also set the following environment variables:

Table 50. Optional z/OS environment variable	
Environment variable	Value
BFG_WTO	One of the following values will enable BFG_WTO : • YES • ON • TRUE One of the following values will disable BFG_WTO. These values are not case sensitive. • NULL • NO • OFF • FALSE Enables z/OS logging. By default, this environment variable is disabled. Messages that are written to the agent event log are also written to the z/OS operator log facility, which allows easier access for automation products when you run an agent from JCL. The routing code is Programmer Information (11) and the descriptor code is Informational (12).

Table 50. Optional z/OS environment variable (continued)

Environment variable	Value
BFG_GROUP_NAME	The mqm file group is typically associated with Managed File Transfer configuration data files and commands. Consequently, all users who are members of the mqm group can access, and make changes to the Managed File Transfer configuration. For more information, see File system permissions for MFT in IBM MQ. For a z/OS system, a file group is a z/OS UNIX filesystem entity, and the mqm file group is not necessarily defined. You can define an alternative, existing z/OS UNIX filesystem group for Managed File Transfer configuration data files by using the BFG_GROUP_NAME environment variable. For example, at the z/OS UNIX shell prompt: <pre>export BFG_GROUP_NAME=FTEGB</pre> which defines group FTEGB to be associated with any subsequently created configuration files for the current z/OS UNIX session. You can set BFG_GROUP_NAME to a blank value, or remove it.

Configuration properties files

A summary of the properties that are used in Managed File Transfer.

- [The MFT coordination.properties file](#)
- [The MFT command.properties file](#)
- [The MFT agent.properties file](#)
- [Logger configuration properties file](#)

Configuring MFT for the z/OS Automatic Restart Manager (ARM)

Managed File Transfer is an ARM enabled application.

Before you begin

For more information about enabling ARM, and defining ARM policies for your system, see [Using the z/OS Automatic Restart Manager \(ARM\)](#).

If you want to use the MFT DB Logger ability to automatically restart and reconnect to a Db2 database, ARM is the only supported restart manager available.

About this task

Using ARM, agents and loggers can be configured for restart by setting the agent/logger properties armELEMTYPE, and armELEMENT. Property armELEMTYPE defines the type of ARM element and property armELEMENT is the name of the element that ARM is to register:

- You can set the agent ELEMTYPE to SYSBFGAG, and armELEMENT can be set to correspond with the agent name.

- You can set the logger ELEMENTYPE to SYSBFGLG, and armELEMENT can be set to correspond with the logger name.

Note: Agents and loggers that are configured for restart by ARM can only be successfully run from a batch job or a started task. Attempts to start the agent or logger from the z/OS UNIX System Services command line directly will fail with an ARM error reason code.

Example

The following example of a restart policy defines agent BFGFT7CAG1 as being dependant on queue manager FT7C:

```
RESTART_ORDER
  LEVEL(3)
  ELEMENT_TYPE(SYSBFGAG, SYSBFGLG)

RESTART_GROUP(GROUP7C)
  ELEMENT(SYSMQMGRFT7C)
  ELEMENT(BFGFT7CAG1)
  RESTART_ATTEMPTS(3,300)
```

Example: Creating JCL for Managed File Transfer agents on z/OS

Use this information to generate some JCL that can be used to create and start an agent on IBM MQ for z/OS.

Copy the sample library

Carry out the following procedure:

1. Make a copy of the library SCSQFCMD (see [“Copy SCSQFCMD to create a JCL library” on page 821](#)) by opening the library.

The majority of the members, those that start with BFGX, BFGY, or BFGZ, are templates that you use to generate the customized JCL for the agent later on.

The important member is BFGCOPY.

2. Open BFGCOPY and replace:

++supplied_library++

with the name of the SCSQFCMD library that was installed as part of the product.

++service-library++

with the name of the library that you want to use for your agent (the target library).

3. Submit the job and you have a new library that you can use.

Edit BFGCUSTM

Carry out the following procedure:

1. Open the new library so that you can edit the BFGCUSTM member (see [“Editing member BFGCUSTM” on page 821](#))

2. Modify all of the parameters in the member that are enclosed within ++ characters, and replace them with the appropriate values. For example, change:

++mq_path++

The path to the z/OS UNIX System Services (z/OS UNIX) Components directory. For example, /mqm/V9R2M0.

Note: There are three instances of this variable to replace.

++bfg_data++

To be the z/OS UNIX directory where your IBM MQ Managed File Transfer for z/OS configuration is to be stored.

++service_type++

To the word AGENT

++agent_name++

To be the name of your agent

Notes:

1. Some of the entries, such as ++options++ required for the CLEAN_AGENT_PROPS, are not needed and so you should remove these.
2. See [“Before you start configuring MFT for z/OS” on page 816](#) for a complete list of all of the parameters in the BFGCUSTM member, along with a description of what values they should have.

Submit the BFGCUSTM JCL

Carry out the following procedure:

1. Submit the job.
2. Exit the library in ISPF.

This is necessary because the BFGCUSTM job is updating the library, and cannot do that while the library is open.

3. When the job completes look at the joblog.

You will see a number of messages, indicating that new members have been created within the library.

Each of these members contains JCL that can be used to perform specific tasks for your agent. See [“z/OS agent and logger command JCL scripts” on page 829](#) for a list of these members, along with the IBM MQ Managed File Transfer commands that they correspond to.

Submit BFGAGCR to create the agent

The new member BFGAGCR contains some JCL that [creates an agent](#) by invoking the **fteCreateAgent** command.

Carry out the following procedure:

1. Open up member BFGAGCR.

You should see that BFGAGCR has been populated with the name of your:

- Agent
- Agent queue manager
- The coordination queue manager for the MFT topology

2. Submit member BFGAGCR.

When the member runs, it:

- Creates the required configuration files for your agent.
- Connects to the agent queue manager, and creates the system queues that the agent needs, using CSQUTIL.
- Registers the agent with the coordination queue manager.

Start the agent by submitting BFGAGST

Carry out the following procedure:

1. Submit the BFGAGST member. See [using the agent](#) for various commands that show you that the agent is working correctly.
2. When the job completes, check the joblog contains the following messages:

```
BFGAG0058I: The agent has successfully initialized.  
BFGAG0059I: The agent has been successfully started.
```

which means that your agent is up, running, and ready to perform managed transfers.

Moving an MFT agent to a new z/OS LPAR

It is sometimes necessary to move an IBM MQ Managed File Transfer for z/OS agent from one LPAR to another, while keeping the agent in the same IBM MQ Managed File Transfer topology with the same coordination and command queue managers. The steps needed to do this depend on how the agent being migrated was originally created.

About this task

Move your IBM MQ Managed File Transfer for z/OS agent in one of the following ways:

- If the agent was originally created using a customized version of the SCSQFCMD library, use the library to recreate it on a new LPAR.
- If the agent was originally created by running z/OS UNIX System Services (z/OS UNIX) commands, use the commands to recreate it on a new LPAR.

Note:

Scheduled transfers and transfer templates are stored on the coordination queue manager for an IBM MQ Managed File Transfer topology. This task assumes that the coordination queue manager is not part of the movement work. In this case, any scheduled transfers and transfer templates associated with the agent being moved remain on the existing coordination queue manager after the move is completed.

Procedure

- Move an agent created using a customized version of the SCSQFCMD library.

If the agent was created using a customized version of the SCSQFCMD library, you can use that library to recreate the IBM MQ Managed File Transfer for z/OS environment, and the agent configuration on the new LPAR. To do this, complete the following steps:

1. Copy the customized version of the library from the original LPAR to the new LPAR.
2. Edit the BFGCUSTM member in the customized version of the library on the new LPAR, and make sure that the parameter values are still valid.
3. Run the BFGCUSTM member on the new LPAR, to create all of the JCL needed to configure the environment and create the agent.
4. Run the BFGCFCR member to define the coordination queue manager to be used by the agent on the new LPAR, and create the directory structure needed to store the IBM MQ Managed File Transfer configuration.
5. Next, run the BFGCMCR member, to define the command queue manager to be used by the agent on the new LPAR.
6. Run the BFGAGCR member to recreate the agent and its configuration.
7. Ensure that the system queues used by the agent exist on the queue manager for that agent.

If the agent being moved has resource monitors associated with it, you need to recreate the monitors on the new agent. To do this, complete the following steps:

1. On the original LPAR, run the BFGMNLI member to export the definitions for the resource monitor associated with the original agent to XML files.
2. Copy the XML files containing the resource monitor definitions to the new LPAR.
3. Use the BFGMNCRS member in the SCSQFCMD library on the new LPAR to import the resource monitor definitions stored in the XML files. This results in the monitors being created on the new agent.

- Move an agent created by running commands in z/OS UNIX.

If the agent was originally created by running z/OS UNIX commands, you can use commands to recreate the agent on a new LPAR. To do this, complete the following steps:

1. Run the `fteSetupCoordination` command on the new LPAR, to define the coordination queue manager to be used by the agent, and create the directory structure needed to store the IBM MQ Managed File Transfer configuration.
2. Run the `fteSetupCommands` command to define the command queue manager to be used by the agent on the new LPAR.
3. Run the `fteCreateAgent` command to recreate the agent and its configuration.
4. Ensure that the system queues used by the agent exist on the queue manager for that agent.

If the agent being moved has resource monitors associated with it, you need to recreate the monitors on the new agent. To do this, complete the following steps:

1. On the original LPAR, run the `fteListMonitors` command, specifying the `-ox` parameter, to export the definitions for the resource monitor, associated with the original agent, to XML files.
2. Copy the XML files containing the resource monitor definitions to the new LPAR.
3. Run the `fteCreateMonitor` command on the new LPAR, specifying the `-ix` parameter, to import the resource monitor definitions stored in the XML files. This results in the monitors being created on the new agent.

► z/OS Planning your MFT infrastructure with IBM MQ for z/OS queue sharing groups

You need to consider the following, if you are using IBM MQ Managed File Transfer (MFT) , when one or more of the agents, command or coordination queue managers are part of an IBM MQ for z/OS queue sharing group.

See [MFT topology overview](#) for a description of agents, command queue managers, and coordination queue managers.

Agent queue managers

Normally an MFT agent connects to a single agent queue manager, and uses local queues that are only accessible by that queue manager. The agent is informed which queue manager to connect to, by providing it with the queue manager name when the agent is first created.

With IBM MQ for z/OS, it is possible to create the agent and replace the queue manager name with the name of a queue sharing group (QSG). This means that the agent can connect to any available queue manager in the QSG to perform file transfers. Should there be a failure of the queue manager that the agent is currently connected to, the agent detects the failure and reconnects to an alternative queue manager in the QSG.

Connecting an agent to a QSG in combination with the highly available agent support provided allows very robust MFT topologies to be created. See [“Highly available agents in Managed File Transfer”](#) on page 862.

For example, in the following figure *Agent1* has been created so that its agent queue manager is a QSG consisting of two queue managers *QM1* and *QM2*. The agent queues have been defined as shared queues, stored in the coupling facility.

This means that the agent can run on either *LPAR 1* or *LPAR 2* and connect to either *QM1* or *QM2*. The files and data sets that the agent reads from, or writes to, are shared, meaning they can be accessed from either LPAR.

In addition, the agent has been configured to be a highly available agent. In the diagram the agent is active in *LPAR 1* and a standby instance of the agent is running in *LPAR 2*.

This topology provides high resilience. Should either the agent running on *LPAR 1* fail, or queue manager *QM1* fail, or *LPAR 1* fail the standby instance of the agent on *LPAR 2* can take over and carry on processing file transfers from the point of failure.

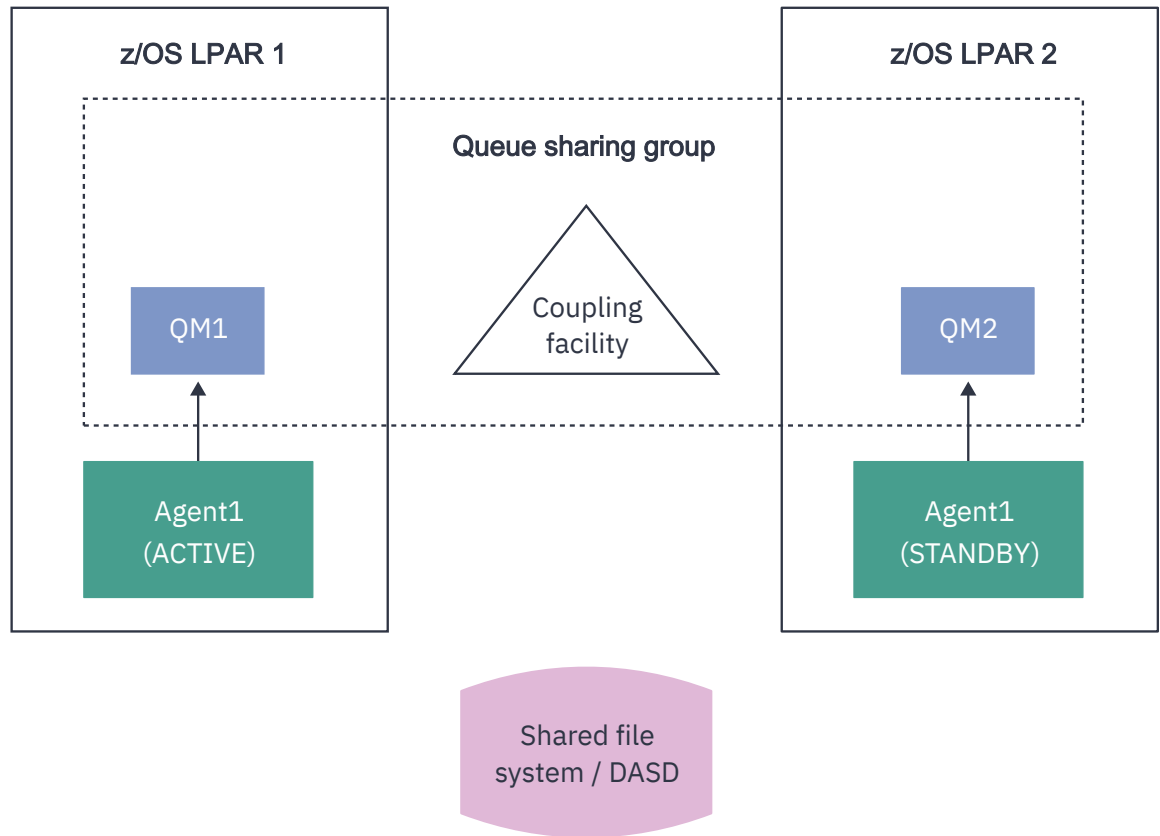


Figure 95. Highly available MFT agent using a queue sharing group

Creating an agent that uses a QSG as the agent queue manager

You create an agent using the `fteCreateAgent` command. When doing this, the name of the queue sharing group is provided for the agent queue manager. For example:

```
fteCreateAgent -agentName Agent1 -agentQMgr QSG1
```

This creates an agent called *Agent1* which uses any queue manager that is a member of QSG *QSG1* as its agent queue manager. In this configuration the agent connects to the agent queue manager using a cross memory (bindings mode) connection which means that the agent and the queue manager must be on the same LPAR. This is exactly like the example shown in figure 1 above.

When you run the **fteCreateAgent** command it generates a set of MQSC commands to create the necessary queues on the agent queue manager.

When the agent queue manager is a QSG, this set of commands needs to be modified so that each queue is created as a shared queue. That is, each queue needs to be created with **QSGDISP(SHARED)** and an appropriate coupling facility structure provided by the **CFSTRUCT** attribute.

The following example shows you how to change the MQSC command for creating the `SYSTEM.FTE.COMMAND.AGENT1` queue as a shared queue by changing the following defaults:

- **QSGDISP(SHARED)**
- **CFSTRUCT(MFTSTRUCT)**

Important: You need to make similar changes to all the other queues that the agent uses.

```
DEFINE QLOCAL(SYSTEM.FTE.COMMAND.AGENT1) +  
  QSGDISP(SHARED) +  
  CFSTRUCT(MFTSTRUCT) +  
  DEFPRTY(0) +  
  DEFSOPT(SHARED) +  
  GET(ENABLED) +  
  INDXTYPE(CORRELID) +  
  MAXDEPTH(5000) +  
  MAXMSGL(4194304) +  
  MSGDLVSQ(PRIORITY) +  
  PUT(ENABLED) +  
  RETINTVL(999999999) +  
  SHARE +  
  NOTRIGGER +  
  USAGE(NORMAL) +  
  REPLACE
```

Creating an agent that uses a QSG as the agent queue manager and connects as a client

Agents can connect to their agent queue manager using a client channel. You can use this approach to allow the agent to run on distributed platforms while connecting to a QSG. If all queue managers in the QSG are licensed for IBM MQ Advanced for z/OS Value Unit Edition, then the agent can also connect to them from a z/OS LPAR that does not have a local queue manager.

This topology is shown in the following figure and allows the agent to take advantage of the resiliency of QSGs. If the queue manager in the QSG that the agent is currently connected to fails, then the agent automatically reconnects to a different member of the QSG and carries on processing.

The sysplex distributor is used to spread the connections from the agent across the available queue managers in the QSG.

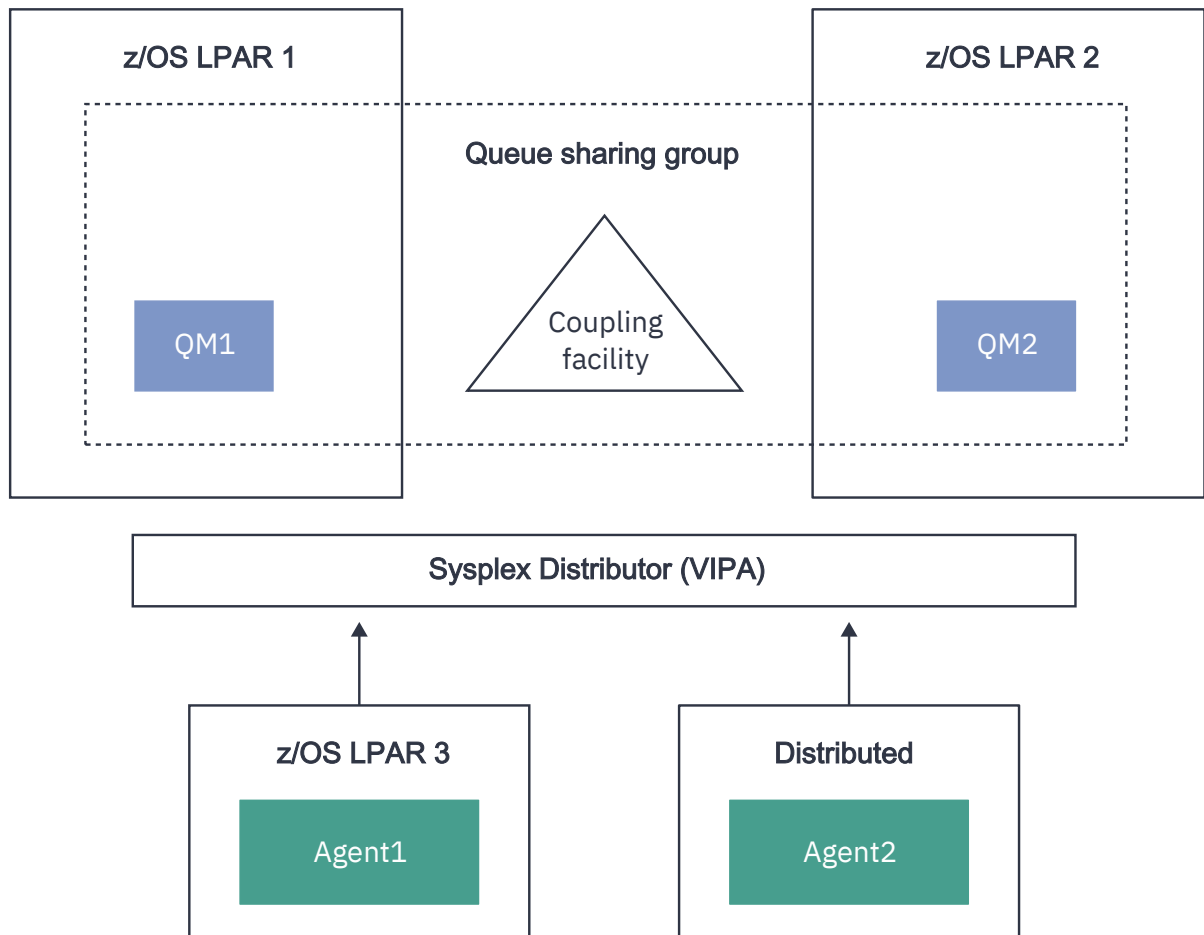


Figure 96. MFT agents connecting to a queue sharing group as a client

In order to make use of this topology, the queue managers in the QSG must each have a server connection channel defined for use by the agent. See [“Connecting a client to a queue sharing group”](#) on page 63 for information on how to do this.

When creating the agent the queue managers need to be configured, so that they can use the channel that is defined to the QSG, and access it through the sysplex distributor. For example:

```
fteCreateAgent -agentName Agent1 -agentQMgr QSG1 -agentQMgrHost vipaAddress
-agentQMgrPort sharedPort -agentQMgrChannel CHANNEL1
```

As previously stated the MQSC commands generated by running the **fteCreateAgent** command must be adjusted to specify QSGDISP(SHARED) and an appropriate coupling facility structure in the CFSTRUCT attribute.

Command queue managers

The MFT command queue manager can be part of a QSG. However, the name of a QSG cannot be used when specifying a command queue manager; you must use a specific queue manager name.

Coordination queue managers

The MFT coordination queue manager can be part of a QSG. However, as with a command queue manager, the name of a QSG cannot be used when specifying a coordination queue manager; you must use a specific queue manager name.

Commands connecting to a QSG

MFT provides a number of commands to manage agents, transfers and agent, command, or coordination queue managers. You can use only those commands that connect to an agent queue manager, if the queue manager is in a QSG.

Following is a list of the commands that connect to the agent queue manager:

- **fteCleanAgent**
- **fteCreateAgent**
- **fteCreateBridgeAgent**
- **fteCreateCDAgent**
- **fteDeleteAgent**

Note that you must provide the name of the queue manager when running other MFT commands.

Using Managed File Transfer for z/OS with the JZOS Java launcher

You can apply the instructions in this topic as an alternative method of using Managed File Transfer in your enterprise, on your IBM MQ for z/OS system.

Overview

Managed File Transfer for z/OS (MFT) uses the standard z/OS installation procedure. An alternative way of running MFT commands is to use JCL and the JZOS Java Launcher.

See [JZOS Batch Launcher and Toolkit](#) for further details.

If your JCL fails to process correctly, see [Common MFT problems with JZOS](#).

Example JCL

```
//JOHNDOEA JOB 1,MSGCLASS=H
// JCLLIB ORDER=(SCEN,MFT,JCL)      (1)
// INCLUDE MEMBER=BFGJCL8           (2)
// DD * (2A)
. ${BFG_PROD}/bin/fteBatch createAgent (3)
export IBM_JAVA_OPTIONS="${BFG_JAVA_OPTIONS} ${BFG_LANG}" (4)
export JZOS_MAIN_ARGS="${BFG_MAIN_ARGS}" (4)
//MAINARGS DD *
-agentName MYAGENT (5)
-f
-agentQMgr MQPD
-p MQPD
/*
```

where:

- (1) Is the location of included JCL statements
- (2) Include the specified JCL member from the location in 1)
- (2A) This extends the //STDENV - see below
- (3) This is the command to be executed, without the leading fte prefix
- (4) These lines are required, they set up information for JZOS
- (5) The parameters to the command
- The BFGJCL8 member (you can select your own name) invokes JZOS. This member has the STEPLIB and other JCL needed to run MFT.

Other JCL you need to include

You should include JCL for the IBM MQ for z/OS libraries, and if you are using the Db2 logger, the Db2 libraries.

For example:

```
//WMQFTE EXEC PGM=JVMLDM86,REGION=0M PARM='+T' (1)
//STPLIB DD DSN=SYS1.SIEALNKE,DISP=SHR (2)
//* MQ libraries
// DD DSN=MQM.V920.SCSQAUTH,DISP=SHR MQ Bindings
// DD DSN=MQM.V920.SCSQANLE,DISP=SHR MQ Bindings
// DD DSN=MQM.V920.SCSQLOAD,DISP=SHR MQ Bindings

//* DB2 libraries
// DD DISP=SHR,DSN=SYS2.DB2.V12.SDSNEXIT.DBCP
// DD DISP=SHR,DSN=SYS2.DB2.V12.SDSNLOAD
// DD DISP=SHR,DSN=SYS2.DB2.V12.SDSNLOAD2
//SYSOUT DD SYSOUT=H
//SYSPRINT DD SYSOUT=H
//STDOUT DD SYSOUT=H
//STDERR DD SYSOUT=H

//STDENV DD DSN=SCEN.MFT.JCL(BFGZENV8),DISP=SHR (3)
```

where:

- (1) Is the name of the JZOS program. Look in SYS1.SIEALNKE for the version on your system. Add ,PARM='+T' to give additional diagnostics.
- (2) This is the data set with the JZOS program.
- (3) This is the member name of a shell script. It defines parameters needed by MFT. See [“Shell script to define MFT”](#) on page 845.

It can be any data set and member. It needs to be last in the file because the JCL job extends this. See 2A in [“Example JCL”](#) on page 844.

Shell script to define MFT

In the [“Other JCL you need to include”](#) on page 844 example, the member BFGZENV8 is used. This is based on the JZOS profile.

You need to know:

- The location where Java is installed
- The location of the IBM MQ for z/OS Java libraries and the MFT libraries.
- A user ID needs to be in a specific group to be considered as an IBM MQ for z/OS administrator. You need the name of this group
- If you are not using English for the messages, you need to know which language to specify.

Example file

```
# This is a shell script that configures
# any environment variables for the Java JVM.
# Variables must be exported to be seen by the launcher.
# Use PARM='+T' and set -x to debug environment script problems
set -x
# . /etc/profile
#
# Java configuration (including MQ Java interface)
#
export _BPXK_AUTOCVT="ON"
export JAVA_HOME="/java/java71_bit64_sr3_fp30/J7.1_64/"
export PATH="/bin:${JAVA_HOME}/bin/classic/"
LIBPATH="/lib:/usr/lib:${JAVA_HOME}/bin"
LIBPATH="${LIBPATH}:${JAVA_HOME}/bin/classic"
LIBPATH=$LIBPATH:"/mqm/V9R2M0/java/lib/"
export LIBPATH

export BFG_JAVA_HOME="${JAVA_HOME}"
export BFG_WTO="YES"
export BFG_GROUP_NAME=MQADM
export BFG_PROD="/mqm/V9R2M0/mqft"
export BFG_CONFIG="/u/johndoe/fteconfig"
```

```
# export BFG_LANG=" -Duser.language=de "  
export BFG_LANG=" "
```

where:

export _BPXK_AUTOCVT="ON"

Is required for Unicode conversion

export JAVA_HOME="/java/java71_bit64/J7.1_64/"

Is the location of the Java directory. Specify the name of the path for Java. This directory contains bin and other directories.

export PATH="/bin:\${JAVA_HOME}/bin/classic/"

Sets up the path statement for Java executable statements

LIBPATH="/lib:/usr/lib:\${JAVA_HOME}/bin"

Sets up the library path for the Java executable statements

LIBPATH="\$LIBPATH:\${JAVA_HOME}/bin/classic"

Adds more Java libraries to the LIBPATH statement.

LIBPATH=\$LIBPATH:"/mqm/V9R2M0/java/lib/"

Adds IBM MQ for z/OS libraries in the library path. Specify the name of your IBM MQ for z/OS libraries in z/OS UNIX System Services.

export LIBPATH

Makes the LIBPATH available to JZOS

export BFG_JAVA_HOME="\${JAVA_HOME}"

Sets the BFG_JAVA_HOME to the value of JAVA_HOME specified above

export BFG_WTO="YES"

Setting BFG_WTO to YES causes messages to be displayed on the joblog using WTO

export BFG_GROUP_NAME=MQADM

User IDs, which are a member of the specified group, are considered IBM MQ for z/OS administrators

export BFG_PROD="/mqm/V9R2M0/mqft"

Is the path where the MFT code is located

export BFG_DATA="/u/johndoe/fteconfig"

Is where the MFT configuration information is stored

export BFG_LANG=" -Duser.language=de "

Is a commented out statement to define the language as German

export BFG_LANG=" "

Specifies the language as the default, English.

The contents of the MFT product in `/lib/messages/BFGNVMessages_*.properties` lists the languages available. The default is to leave the value blank, which means that English is used.

Related tasks

[“Configuring Managed File Transfer for z/OS” on page 815](#)

Managed File Transfer for z/OS requires customization to enable the component to operate correctly.

[Planning for Managed File Transfer](#)

IBM i Configuring MFT on IBM i

To start using Managed File Transfer after you have installed it, you must complete some configuration for your coordination queue manager and agent.

About this task

After you have installed, you must run the configuration scripts provided by Managed File Transfer for new coordination queue managers and new agents before you can use the coordination queue managers and agents to transfer files. You must then start the agents you have created.

Procedure

1. For all new coordination queue managers: run the MQSC commands in the `coordination_qmgr_name.mqsc` file against the coordination queue manager. If the coordination queue manager is not on the same computer as the installation, copy the MQSC script file to the computer where the queue manager is located and then run the script.
 - a) From an IBM i command line, start qshell using the following command: `CALL QSHELL`
 - b) Change to the following directory: `/QIBM/UserData/mqm/mqft/config/coordination_qmgr_name`
 - c) Issue the following command, replacing `coordination_qmgr_name` with the name of your queue manager:

```
/QSYS.LIB/QMQM.LIB/RUNMQSC.PGM coordination_qmgr_name < coordination_qmgr_name.mqsc
```

You can configure the coordination queue manager manually instead. For more information, see [“Configuring the coordination queue manager for MFT”](#) on page 851.

2. For all new agents: run the MQSC commands in the `agent_name_create.mqsc` file against the agent queue manager.

If the agent queue manager is not on the same computer as the agent, copy the MQSC script file to the computer where the queue manager is located and then run the script.

 - a) From an IBM i command line, start qshell using the following command: `CALL QSHELL`
 - b) Change to the following directory: `/QIBM/UserData/mqm/mqft/config/agent_qmgr_name/agents`
 - c) Issue the following command, replacing `agent_qmgr_name` with the name of your agent queue manager and replacing `agent_name` with the name of your agent:

```
/QSYS.LIB/QMQM.LIB/RUNMQSC.PGM agent_qmgr_name < agent_name_create.mqsc
```

You can configure the agent queue manager manually instead. For more information, see [“Configuring MFT agent queue managers”](#) on page 858.

3. If you have not already started the QMFT subsystem as part of the installation, from the IBM i command line, start the QMFT subsystem using the following command: `STRSBS SBSD(QMQMMFT/QMFT)`, or `STRSBS QMQMMFT/QMFT`
4. Start your new agents using the **fteStartAgent** command.
 - a) From an IBM i command line, start qshell using the following command: `CALL QSHELL`
 - b) Change to the following directory: `/QIBM/ProdData/mqm/bin`
 - c) Issue the following command, replacing `AGENT` with the name of your agent:

```
./fteStartAgent AGENT
```

What to do next

You are recommended to set up sandboxes to limit the areas of the file system that an agent can access. This feature is described in [Working with MFT agent sandboxes](#).

Related concepts

[“Configuring MFT for first use”](#) on page 848

You must perform some configuration tasks for Managed File Transfer agents and queue managers once, the first time you want to use them.

Configuring MFT for first use

You must perform some configuration tasks for Managed File Transfer agents and queue managers once, the first time you want to use them.

Related concepts

[“Connecting to IBM MQ” on page 849](#)

All network communication with IBM MQ queue managers, including communication related to Managed File Transfer, involves IBM MQ channels. An IBM MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

[“Configuring a multi-instance queue manager to work with MFT” on page 854](#)

IBM WebSphere MQ 7.0.1 onwards supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Related tasks

[“Configuring MFT network queue managers” on page 850](#)

If your Managed File Transfer network includes more than one IBM MQ queue manager, these IBM MQ queue managers must be able to remotely communicate with each other.

[“Configuring MFT agent queue managers” on page 858](#)

After installation, run the *agent_name_create.mqsc* script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name* directory to perform the necessary configuration for the agent queue manager. However, if you want to do this configuration manually, complete these steps on the agent queue manager.

[“Configuring the coordination queue manager for MFT” on page 851](#)

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name.mqsc* script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

[“Creating an MFT Agent or Logger command data set” on page 814](#)

You can create a PDSE data set of commands from the Managed File Transfer command template data set for a specific Managed File Transfer Agent or Managed File Transfer Logger for a specific coordination.

[“Updating an existing MFT Agent or Logger command data set on z/OS” on page 825](#)

You can update an Managed File Transfer command PDSE library data set that is created from the Managed File Transfer command template data set.

Related reference

[MFT Agent queue settings](#)

[MFT system queues and the system topic](#)

[“Retaining MFT log messages” on page 856](#)

Managed File Transfer sends file transfer progress and log information to the coordination queue manager. The coordination queue manager publishes this information to any matching subscriptions to the `SYSTEM.FTE` topic. If there are no subscriptions, this information is not retained.

Connecting to IBM MQ

All network communication with IBM MQ queue managers, including communication related to Managed File Transfer, involves IBM MQ channels. An IBM MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

Managed File Transfer and channels

Managed File Transfer uses MQI channels to connect agents in client mode to their agent queue managers, and to connect command applications (for example, **fteCreateTransfer**) to their command and coordination queue managers. In the default configuration, these connections are made using a `SVRCONN` channel called `SYSTEM.DEF.SVRCONN`, which exists by default on all queue managers. Because of these defaults, you do not need to alter any MQI channels for a basic Managed File Transfer installation.

There are six types of message channel end points, but this topic covers only sender-receiver pairs. See [Distributed queuing components](#) for information about other channel combinations.

Required message paths

IBM MQ messages can travel over message channels only, so you must ensure that channels are available for all message paths required by Managed File Transfer. These paths do not need to be direct; messages can travel through intermediate queue managers if required. This topic covers only direct point-to-point communication. See [How to get to the remote queue manager](#) for more information about these options.

The communication paths used by Managed File Transfer are as follows:

Agent to agent

Any two agents that files are transferred between require bidirectional communication between their associated queue managers. Because this path carries the bulk data, consider making the path as short, fast, or cheap as possible according to your needs.

Agent to coordination

Log messages from the agents that participate in a transfer must be able to reach the coordination queue manager.

Command to agent

Any queue manager that command applications or the IBM MQ Explorer (using the command queue manager) connect to must be able to send messages to the queue managers of the agents that those command applications are used to control. To enable feedback messages to be shown by the commands, use a bidirectional connection.

For more information, see *Verifying an IBM MQ installation* for the platform, or platforms, that your enterprise use.

Related concepts

[“Configuring a multi-instance queue manager to work with MFT” on page 854](#)

IBM WebSphere MQ 7.0.1 onwards supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Related tasks

[“Configuring MFT network queue managers” on page 850](#)

If your Managed File Transfer network includes more than one IBM MQ queue manager, these IBM MQ queue managers must be able to remotely communicate with each other.

[“Configuring the coordination queue manager for MFT” on page 851](#)

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name.mqsc* script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Configuring MFT network queue managers

If your Managed File Transfer network includes more than one IBM MQ queue manager, these IBM MQ queue managers must be able to remotely communicate with each other.

About this task

There are two ways to configure your queue managers to be able to communicate with each other:

- By setting up an IBM MQ queue manager cluster.

For information about IBM MQ queue manager clusters and how to configure them, see [“Configuring a queue manager cluster” on page 309](#).

- By setting up channels between the queue managers, which is described as follows:

Setting up channels between queue managers

Set up the following message channels between your queue managers:

- From the agent queue manager to the coordination queue manager
- From the command queue manager to the agent queue manager.
- From the agent queue manager to the command queue manager (to enable feedback messages to be shown by the commands).
- From the command queue manager to the coordination queue manager
- From the agent queue manager to any other agent queue manager in the Managed File Transfer network

If you need further information about how to set up this communication, start with this information: [Administering remote IBM MQ objects using MQSC](#).

Some suggested example steps are:

Procedure

1. Create a transmission queue on the IBM MQ queue manager with the same name as the coordination queue manager.

You can use the following MQSC command:

```
DEFINE QLOCAL(coordination-qmgr-name) USAGE(XMITQ)
```

2. On the IBM MQ queue manager, create a sender channel to the Managed File Transfer coordination queue manager.

The name of the transmission queue created in the previous step is a required parameter for this channel.

For agents on Managed File Transfer for IBM MQ, messages are published with a blank format.

You can use the following MQSC command:

```
DEFINE CHANNEL(channel-name) CHLTYPE(SDR) CONNAME('coordination-qmgr-host(coordination-qmgr-port)')  
XMITQ(coordination-qmgr-name) CONVERT(NO)
```

Note: Set CONVERT(NO), only if required.

3. On the Managed File Transfer coordination queue manager, create a receiver channel to the IBM MQ queue manager. Give this receiver channel the same name as the sender channel on the IBM MQ queue manager.

You can use the following MQSC command:

```
DEFINE CHANNEL(channel-name) CHLTYPE(RCVR)
```

What to do next

Next, follow the configuration steps for your coordination queue manager: [“Configuring the coordination queue manager for MFT” on page 851](#).

Related concepts

[“Connecting to IBM MQ” on page 849](#)

All network communication with IBM MQ queue managers, including communication related to Managed File Transfer, involves IBM MQ channels. An IBM MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

[“Configuring a multi-instance queue manager to work with MFT” on page 854](#)

IBM WebSphere MQ 7.0.1 onwards supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Related tasks

[“Configuring the coordination queue manager for MFT” on page 851](#)

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name.mqsc* script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Configuring the coordination queue manager for MFT

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name.mqsc* script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

About this task

Procedure

1. Create a local queue named SYSTEM.FTE.
2. Add the SYSTEM.FTE queue to the SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist.
3. Create a topic named SYSTEM.FTE with a topic string of SYSTEM.FTE.
4. Ensure the Non-persistent message delivery (NPMSGDLV) and Persistent messages delivery (PMSGDLV) attributes of the SYSTEM.FTE topic are set to ALLAVAIL.
5. Ensure the Publish/Subscribe mode (PSMODE) attribute of the coordination queue manager is set to ENABLED.

What to do next

If you run the `strmqm -c` command on a queue manager that has been configured as a coordination queue manager, the command deletes the change made in [step 2](#) (adding the SYSTEM.FTE queue to the SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist). This is because `strmqm -c` re-creates the default IBM MQ objects and reverses the Managed File Transfer changes. Therefore, if you have started the queue manager with `strmqm -c`, complete either one of the following steps:

- Run the *coordination_qmgr_name.mqsc* script on the queue manager again.
- Repeat [step 2](#).

Related concepts

[“Connecting to IBM MQ” on page 849](#)

All network communication with IBM MQ queue managers, including communication related to Managed File Transfer, involves IBM MQ channels. An IBM MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

[“Configuring a multi-instance queue manager to work with MFT” on page 854](#)

IBM WebSphere MQ 7.0.1 onwards supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Related tasks

[“Configuring MFT network queue managers” on page 850](#)

If your Managed File Transfer network includes more than one IBM MQ queue manager, these IBM MQ queue managers must be able to remotely communicate with each other.

Related reference

[fteSetupCoordination](#)

Creating an IBM MQ File Transfer structure

You can configure a Managed File Transfer structure, based on a single agent connected to a queue manager on the same machine.

About this task

The MFT configuration is stored in a file structure under the IBM MQ DataPath, on the machine that the agent will be located.

The following sample configuration is for an MFT in IBM MQ 8.0 queue manager named SAMPLECOORD (with security disabled), and a single MFT agent named SAMPLEAGENT:

```
+--- config
      +--- SAMPLECOORD
            +--- command.properties
            +--- coordination.properties
            +--- SAMPLECOORD.mqsc
            +--- agents
                  +--- SAMPLEAGENT
                        +--- agent.properties
                        +--- SAMPLEAGENT_create.mqsc
                        +--- SAMPLEAGENT_delete.mqsc

+---logs
      +--- SAMPLECOORD
            +--- agents
                  +--- SAMPLEAGENT
                        +--- logs
```

This example assumes that queue manager security has been disabled. The following commands, run in **runmqsc**, will disable security after the queue manager is restarted:

```
runmqsc queue manager
alter qmgr CONNAUTH(NONE);
alter qmgr CHLAUTH(DISABLED);
end;
```

For configuration with security enabled in MFT in IBM MQ 8.0 or later, **CONNAUTH** requires all MFT commands that connect with a queue manager to supply user ID and password credentials. You can apply the additional parameters **-mquserid** and **-mqpassword** for each command, or define a `MQMFTCredentials.xml` file. The following sample credential file defines the user ID of `fteuser`, for which the password of `MyPassword` is to be used when connecting to the queue manager `SAMPLECOORD`:

```
<tns:mqmftCredentials xmlns:tns="http://wmqfte.ibm.com/MQMFTCredentials"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/MQMFTCredentials MQMFTCredentials.xsd">
```



```
<tns:qmgr mqPassword="MyPassword" MyUserId="fteuser" name="SAMPLCOORD"/>
</tns:mqmftCredentials>
```

For more information, see [MFT and IBM MQ connection authentication](#).

Notes:

- To locate your MFT configuration directory, use the **fteDisplayVersion -v** command.
-  For z/OS users, the MQMFTCredential.xml file can be located as a member in a partitioned data set with variable record format (RECFM=V), or undefined record format (RECFM=U).
- For configuration with security enabled, add the following parameter to the steps below to associate the credentials with the relevant queue manager: **-F full_credential_file_path**.
- The clear text password in the MQMFTCredential.xml can be obfuscated using the following command:

```
fteObfuscate -f full_file_path_to_MQMFTCredentials.xml
```

Procedure

1. Create a coordination queue manager.

A coordination queue manager is a single queue manager, used to receive all transfer log and status information from its agents. Run the following command:

```
fteSetupCoordination -coordinationQMGr coordination_qmgr_name
```

This creates the basic top level configuration, and create an IBM MQ script file to call *coordination_qmgr_name.mqsc*.

The configuration then needs to be loaded into the queue manager, by running the following IBM MQ command:

```
runmqsc queue_manager_name < coordination_qmgr_name.mqsc
```

Note: For TCP client connection to a queue manager, you can use:

```
fteSetupCoordination -coordinationQMGr coordination_qmgr_name
                    -coordinationQMGrHost coordination_qmgr_host -coordinationQMGrPort coordination_qmgr_port
                    -coordinationQMGrChannel coordination_qmgr_channel
```

For the created *coordination_qmgr_name.mqsc*, you will need to run the **runmqsc** command on the same machine that the coordination queue manager is running on.

2. Create the command queue manager.

A command queue manager is a single queue manager which has been pre-configured so that the IBM MQ infrastructure can route MFT requests to the relevant agent. Run the following command:

```
fteSetupCommands -connectionQMGr Command QM Name -p Coordination QM Name
```

This creates a *command.properties* file in the coordination directory. Note that the **-p** is optional, and not required if the commands are being set up for the default coordination.

Note: For TCP client connection to a queue manager, you can use:

```
fteSetupCommands -p coordination_qmgr_name -commandQMGr connection_qmgr_name
                    -commandQMGrHost connection_qmgr_host -commandQMGrPort connection_qmgr_port
                    -commandQMGrChannel connection_qmgr_channel
```

3. Create the agent.

An agent is an application that can send and receive files. Run the following command:

```
fteCreateAgent -p coordination_qmgr_name -agentName agent_name -agentQMGr agent_qmgr_name
```

This creates the agent configuration under the coordination, and creates an IBM MQ script file to call *agent_name.mqsc* in the agent's configuration directory.

Run the following IBM MQ command to load the IBM MQ script file into the queue manager:

```
runmqsc agent_qmgr_name < agent_name_create.mqsc file
```

Note: For TCP client connection to a queue manager, you can use:

```
fteCreateAgent -p coordination_qmgr_name -agentName agent_name -agentQMGr agent_qmgr_name  
-agentQMGrHost agent_qmgr_host -agentQMGrPort agent_qmgr_port -agentQMGrChannel  
agent_qmgr_channel
```

4. Start the agent.

Run the following command:

```
fteStartAgent -p coordination_qmgr_name agentName
```

The agent starts in the background, and the command prompt is returned. To check that the agent is running, run the following command:

```
ftelListAgents -p coordination_qmgr_name
```

This shows the status of the agents. If the agent is running successfully, it is reported as in the READY state.

Results

A basic MFT infrastructure is ready to use, and you can now use the **fteCreateTransfer** command to request a transfer. Alternatively if the IBM MQ Explorer is available, use the MFT plug-ins to create and monitor transfers.

More agents can be added to the configuration by repeating the Step 3: Create the agent. If the TCP client connection is used, these can be on different machines. For different machines the **fteSetupCoordination** and **fteSetupCommands** commands must be repeated for each machine, however the mqsc scripts would not need to be run.

More complex configurations can have separate queue managers for coordination and each agent. In these cases the various queue managers will need to be connected together.

Related concepts

[What to do if your MFT agent is not listed by the **ftelListAgents** command](#)

Related reference

[fteSetupCoordination](#)

[fteSetupCommands: create the MFT command.properties file](#)

[fteCreateAgent](#)

[fteObfuscate](#): encrypt sensitive data

[MFT credentials file format](#)

[The MFTagent.properties file](#)

Configuring a multi-instance queue manager to work with MFT

IBM WebSphere MQ 7.0.1 onwards supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. Managed File Transfer supports

connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Configuring a multi-instance queue manager

Important: For information about configuring an IBM MQ multi-instance queue manager, see [“Multi-instance queue managers”](#) on page 540. Ensure that you have read this information before attempting to configure a multi-instance queue manager to work with Managed File Transfer.

Using a multi-instance queue manager as an agent queue manager

To enable an agent to connect to both the active and standby instance of your multi-instance queue manager, add the `agentQMGrStandby` property to the agent's `agent.properties` file. The `agentQMGrStandby` property defines the host name and the port number used for client connections for the standby queue manager instance. The value of the property must be given in MQ CONNAME format, that is, `host_name(port_number)`.

The `agentQMGr` property specifies the name of the multi-instance queue manager. The `agentQMGrHost` property specifies host name for the active queue manager instance and the `agentQMGrPort` property specifies the port number for the active queue manager instance. The agent must connect in client mode to both the active and the standby instance of the multi-instance queue manager.

See [The MFT `agent.properties` file](#) for more information.

This example shows the contents of the `agent.properties` file for AGENT1 that connects to a multi-instance queue manager called QM_JUPITER. The active instance of QM_JUPITER is on the system host1 and uses the port number 1414 for client connections. The standby instance of QM_JUPITER is on the system host2 and uses port number 1414 for client connections.

```
agentName=AGENT1
agentDesc=
agentQMGr=QM_JUPITER
agentQMGrPort=1414
agentQMGrHost=host1
agentQMGrChannel=SYSTEM.DEF.SVRCONN
agentQMGrStandby=host2(1414)
```

Using a multi-instance queue manager as the coordination queue manager

To enable connections to both the active and standby instance of your multi-instance coordination queue manager, add the `coordinationQMGrStandby` property to all the `coordination.properties` files in your Managed File Transfer topology.

See [The MFT `coordination.properties` file](#) for more information.

This example shows the contents of a `coordination.properties` file that specifies the connection details to a multi-instance coordination queue manager called QM_SATURN. The active instance of QM_SATURN is on the system `coordination_host1` and uses the port number 1420 for client connections. The standby instance of QM_SATURN is on the system `coordination_host2` and uses the port number 1420 for client connections.

```
coordinationQMGr=QM_SATURN
coordinationQMGrHost=coordination_host1
coordinationQMGrPort=1420
coordinationQMGrChannel=SYSTEM.DEF.SVRCONN
coordinationQMGrStandby=coordination_host2(1420)
```

The Managed File Transfer stand-alone logger must always connect to its queue manager in bindings mode. When using the stand-alone logger with a multi-instance coordination queue manager connect stand-alone logger, in bindings mode, to a different queue manager. The steps to do this are described in [“Alternative configurations for an MFT stand-alone logger”](#) on page 890. You must define the channels between the stand-alone logger's queue manager and the coordination queue manager with

the host name and port number of both instances of the multi-instance coordination queue manager. For information on how to do this, see [“Multi-instance queue managers”](#) on page 540.

The Managed File Transfer plug-in for IBM MQ Explorer connects to the coordination queue manager in client mode. If the active instance of the multi-instance coordination queue manager fails the standby instance of the coordination queue manager becomes active and the plug-in reconnects.

The Managed File Transfer commands **fteList*** and **fteShowAgentDetails** connect directly to the coordination queue manager. If the active instance of the multi-instance coordination is unavailable these commands will attempt to connect to the standby instance of the coordination queue manager.

Using a multi-instance queue manager as the command queue manager

To enable connections to both the active and standby instance of your multi-instance command queue manager, add the `connectionQMGrStandby` property to all the `command.properties` files in your Managed File Transfer topology.

See [The MFT `command.properties` file](#) for more information.

This example shows the contents of a `command.properties` file that specifies the connection details to a multi-instance command queue manager called QM_MARS. The active instance of QM_MARS is on the system `command_host1` and uses the port number 1424 for client connections. The standby instance of QM_MARS is on the system `command_host2` and uses the port number 1424 for client connections.

```
connectionQMGr=QM_SATURN
connectionQMGrHost=command_host1
connectionQMGrPort=1424
connectionQMGrChannel=SYSTEM.DEF.SVRCONN
connectionQMGrStandby=command_host2(1424)
```

Related concepts

[“Connecting to IBM MQ”](#) on page 849

All network communication with IBM MQ queue managers, including communication related to Managed File Transfer, involves IBM MQ channels. An IBM MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

Related tasks

[“Configuring MFT network queue managers”](#) on page 850

If your Managed File Transfer network includes more than one IBM MQ queue manager, these IBM MQ queue managers must be able to remotely communicate with each other.

[“Configuring the coordination queue manager for MFT”](#) on page 851

After running the **fteSetupCoordination** command, run the `coordination_qmgr_name.mqsc` script in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name` directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Retaining MFT log messages

Managed File Transfer sends file transfer progress and log information to the coordination queue manager. The coordination queue manager publishes this information to any matching subscriptions to the `SYSTEM.FTE` topic. If there are no subscriptions, this information is not retained.

Ways of ensuring that information is retained

If transfer progress or log information is significant to your business, you must take one of the following steps to ensure that the information is retained:

- Use the Managed File Transfer database logger to copy messages published to the `SYSTEM.FTE/Log` topic to an Oracle or Db2 database.

- Define a subscription to the SYSTEM.FTE topic, which stores publications on an IBM MQ queue. Define this subscription before transferring any file transfers to ensure that all progress and log messages are retained on the queue.
- Write an application that uses the message queue interface (MQI) or IBM MQ JMS to create a durable subscription and process the publications that are delivered to the subscription. This application must be in operation before any files are transferred to ensure that the application receives all progress and log messages.

Each of these approaches is described in more detail in the sections that follow.

Do not rely on the IBM MQ Explorer plug-in to retain log information.

Using the Managed File Transfer database logger to retain log messages

The database logger is an optional component of Managed File Transfer that you can use to copy log information in to a database for analysis and auditing purposes. The database logger is a stand-alone Java application that you install on a system that hosts the coordination queue manager and the database. For more information about the database logger, see [“Configuring an MFT logger”](#) on page 868.

Retaining progress and log messages by using the IBM MQ Explorer plug-in

When an instance of the IBM MQ Explorer plug-in is first started, the instance creates a durable subscription on the coordination queue manager. This durable subscription is used to collect the information that is displayed in the **Transfer Log** and **Current Transfer Progress** views.

The name of the durable subscription is prefixed to show that the subscription was created by the IBM MQ Explorer MFT plug-in, the host name, and the name of the user. For example MQExplorer_MFT_Plugin_HOST_TJWatson.

This prefix is added in case an administrator wants to delete a durable subscription that is no longer in active use by an instance of the IBM MQ Explorer plug-in.

Using a durable subscription on the coordination queue manager can cause messages to build up on the SYSTEM.MANAGED.DURABLE queues. If you have a high-volume Managed File Transfer network, use the IBM MQ Explorer plug-in infrequently, or both, this message data can fill the local file system.

To stop this from happening, specify that the IBM MQ Explorer plug-in use a non-durable subscription to the coordination queue manager. Perform the following steps in your IBM MQ Explorer:

1. Select **Window > Preferences > MQ Explorer > Managed File Transfer**
2. From the **Transfer Log subscription type** list, choose NON_DURABLE.

Storing publications on an IBM MQ queue

To store log or progress messages on an IBM MQ queue, configure a subscription on the coordination queue manager that forwards messages to this queue. For example, to forward all log messages to a queue named LOG.QUEUE, submit the following MQSC command:

```
define sub(MY.SUB) TOPICSTR('Log/#') TOPICOBJ(SYSTEM.FTE) DEST(LOG.QUEUE)WSHEMA(TOPIC)
```

After the log messages have been forwarded to an IBM MQ queue, they are persisted on the queue until they are processed by an IBM MQ application that uses the queue.

Writing applications that manage a durable subscription to the SYSTEM.FTE topic

You can write applications that manage their own durable subscriptions to the SYSTEM.FTE topic by using one of the application programming interfaces supported by IBM MQ. These applications can receive IBM MQ queue or log messages and act on them appropriately for your business needs.

For more information about the available application programming interfaces, see [Developing applications](#).

Configuring MFT agent queue managers

After installation, run the *agent_name_create.mqsc* script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name/agents/agent_name* directory to perform the necessary configuration for the agent queue manager. However, if you want to do this configuration manually, complete these steps on the agent queue manager.

Procedure

1. Create the agent operation queues.

These queues are named:

- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

For information about the queue parameters and how the queues are used, see [MFT Agent queue settings](#).

2. Create the agent authority queues.

These queues are named:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*

For information about the queue parameters and how the queues are used, see [MFT Agent queue settings](#).

What to do next

For information about creating and configuring a protocol bridge agent, see [fteCreateBridgeAgent \(create and configure an MFT protocol bridge agent\)](#) and [Configuring a protocol bridge for an FTPS server](#).

Related concepts

[“Connecting to IBM MQ” on page 849](#)

All network communication with IBM MQ queue managers, including communication related to Managed File Transfer, involves IBM MQ channels. An IBM MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

[“Configuring a multi-instance queue manager to work with MFT” on page 854](#)

IBM WebSphere MQ 7.0.1 onwards supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Related tasks

[Collecting information for MFT agent problems](#)

[Troubleshooting MFT agent status problems](#)

[“Configuring MFT network queue managers” on page 850](#)

If your Managed File Transfer network includes more than one IBM MQ queue manager, these IBM MQ queue managers must be able to remotely communicate with each other.

[“Configuring the coordination queue manager for MFT” on page 851](#)

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name.mqsc* script in the *MQ_DATA_PATH/mqft/config/coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Related reference

[MFT Agent queue settings](#)

[fteSetupCoordination](#)

Configuring an MFT agent for multiple channels in a cluster

If you want to use IBM MQ multi-channel support in a clustered configuration, first set the **agentMultipleChannelsEnabled** property to `true` and then complete the steps in this topic.

About this task

In a cluster, multi-channel support is enabled by IBM MQ definitions on the queue manager of the destination agent only.

You must complete the steps in this topic in addition to the standard IBM MQ configuration steps required for a Managed File Transfer agent, which are listed in [“Configuring MFT for first use”](#) on page 848.

The following configuration examples use **runmqsc** commands.

Procedure

1. Define a cluster-receiver channel for each channel that you want to use. For example, if you are using two channels:

```
DEFINE CHANNEL(TO.DESTQMGRNAME_1) CHLTYPE(CLUSRCVR) CLUSTER(MFTCLUSTER)
DEFINE CHANNEL(TO.DESTQMGRNAME_2) CHLTYPE(CLUSRCVR) CLUSTER(MFTCLUSTER)
```

where:

- *DESTQMGRNAME* is the name of the queue manager of the destination agent.
- *MFTCLUSTER* is the name of the IBM MQ cluster.

You are recommended to use the *MFTCLUSTER.DESTQMGRNAME_n* naming convention for channels, but this convention is not mandatory.

2. Define a queue manager alias corresponding to each channel. For example:

```
DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_1) RQMNAME(DESTQMGRNAME) CLUSTER(MFTCLUSTER)
DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_2) RQMNAME(DESTQMGRNAME) CLUSTER(MFTCLUSTER)
```

You must use the *SYSTEM.FTE.DESTQMGRNAME_n* naming convention for [queue manager aliases](#) because the sending agent searches for queue manager aliases of this format. The numbers that you use for *n* must start at 1 and be consecutive. You must make the definitions cluster-wide so that they are available on the source agent's queue manager.

For both the source agent and destination agent to correctly determine the number of queue manager aliases, do **not** define a default XMITQ for the queue manager. A default XMITQ or transmission queue is the queue on a queue manager used when the **XMITQ** parameter is not defined anywhere. For more information, see [Queues used for specific purposes by IBM MQ](#).

Related tasks

[“Configuring an MFT agent for multiple channels: non-clustered”](#) on page 860

If you want to use IBM MQ multi-channel support in a non-clustered configuration, first set the **agentMultipleChannelsEnabled** property to `true` and then complete the steps in this topic.

Related reference

[The MFT agent.properties file](#)

Configuring an MFT agent for multiple channels: non-clustered

If you want to use IBM MQ multi-channel support in a non-clustered configuration, first set the `agentMultipleChannelsEnabled` property to `true` and then complete the steps in this topic.

About this task

In a non-clustered configuration, multi-channel support is enabled by IBM MQ definitions on the queue manager of both the source agent and the destination agent.

You must complete the steps in this topic in addition to the standard IBM MQ configuration steps required for a Managed File Transfer agent, which are listed in [“Configuring MFT for first use” on page 848](#).

The following steps assume that sender-receiver channels are being used to communicate between the source and destination queue managers.

The following configuration examples use **runmqsc** commands.

Procedure

1. On the destination agent's queue manager, define a receiver channel for each channel that you want to use. For example, if you are using two channels:

```
DEFINE CHANNEL(TO.DESTQMGRNAME_1) CHLTYPE(RCVR) TRPTYPE(TCP)
DEFINE CHANNEL(TO.DESTQMGRNAME_2) CHLTYPE(RCVR) TRPTYPE(TCP)
```

where: `DESTQMGRNAME` is the name of the queue manager of the destination agent.

You are recommended to use the `TO.DESTMGRNAME_n` naming convention for channels, but this convention is not mandatory. The receiver channel names must match the corresponding sender channels on the source agent's queue manager.

2. On the source agent's queue manager, define a transmission queue for each channel that you want to use. For example, if you are using two channels:

```
DEFINE QLOCAL(DESTQMGRNAME_1) USAGE(XMITQ)
DEFINE QLOCAL(DESTQMGRNAME_2) USAGE(XMITQ)
```

You are recommended to use the `DESTMGRNAME_n` naming convention for transmission queues, but this convention is not mandatory. The transmission queues you define are referenced from the sender channel definitions and the queue manager alias definitions in the following steps.

3. On the source agent's queue manager, define a sender channel for each channel that you want to use. For example, if you are using two channels:

```
DEFINE CHANNEL(TO.DESTQMGRNAME_1) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(DESTHOST:port)
XMITQ(DESTQMGRNAME_1)
DEFINE CHANNEL(TO.DESTQMGRNAME_2) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(DESTHOST:port)
XMITQ(DESTQMGRNAME_2)
```

You are recommended to use the `TO.DESTMGRNAME_n` naming convention for the channels, but this convention is not mandatory. The sender channel names must match the corresponding receiver channels on the destination agent's queue manager.

4. On the source agent's queue manager, define a queue manager alias corresponding to each channel. For example:

```
DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_1) RQMNAME(DESTQMGRNAME) XMITQ(DESTQMGRNAME_1)
DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_2) RQMNAME(DESTQMGRNAME) XMITQ(DESTQMGRNAME_2)
```

You must use the `SYSTEM.FTE.DESTQMGRNAME_n` naming convention for the queue manager aliases because the sending agent searches for queues manager aliases of this format. The numbers that you use for *n* must start at 1 and be consecutive.

For the agent to correctly determine the number of queue manager aliases, do **not** define a default `XMITQ` for the queue manager. A default `XMITQ` or transmission queue is the queue on a queue

manager used when the **XMITQ** parameter is not defined anywhere. For more information, see [Queues used for specific purposes by IBM MQ](#).

Related tasks

[“Configuring an MFT agent for multiple channels in a cluster” on page 859](#)

If you want to use IBM MQ multi-channel support in a clustered configuration, first set the **agentMultipleChannelsEnabled** property to `true` and then complete the steps in this topic.

Related reference

[The MFT agent.properties file](#)

Configuring MFT agents with MSCS

Managed File Transfer (MFT) agent Microsoft Cluster Service (MSCS) setup is supported, if the platform is one supported by MFT and running one of the versions of Windows.

About this task

This task describes two scenarios that you can follow to achieve failover of an MFT agent:

- Scenario 1: Configuring the agent as an MSCS resource.
- Scenario 2: Configuring the agent queue manager and the agent as MSCS resources.

Procedure

Scenario 1: Configuring the agent as an MSCS resource

- To configure the agent as an MSCS resource, complete the following steps:
 - a) Install Managed File Transfer locally on each machine in the cluster.
See [Installing Managed File Transfer](#).
 - b) Create the agent on the primary machine in the cluster.
The agent should be configured to connect to the agent queue manager using the CLIENT transport. Ensure that you create all of the objects on the queue manager for this agent. For information on how to do this, see [Setting up the agent](#).
 - c) Modify the agent to run as a Windows service, and configure it so that it does not start automatically when Windows restarts by setting the **Startup Type** field for the agent service in the Windows Services tool to Manual.
For more information, see [Starting an MFT agent as a Windows service](#).
 - d) Repeat step [“2” on page 861](#) and step [“3” on page 861](#) of Scenario 1 on the secondary machine.
This ensures that the file structure for logs, properties, and so on, exists on the other machine in the cluster. Note that there is no need to create the queue manager objects as in step [“2” on page 861](#).
 - e) On the primary machine, add the agent as a 'Generic Service' under MSCS control.
To do this:
 - a. Right-click the cluster and select the **Role -> Add Resource -> 'Generic Service'**.
 - b. From the list of Windows services, select the agent service and complete the configuration wizard by clicking **Next**.The agent service is now added as an MSCS resource. If a failover occurs, then the agent service will be started on the other machine.

Scenario 2: Configuring the agent queue manager and the agent as MSCS resources

- To configure the agent queue manager and the agent as MSCS resources, complete the following steps:
 - a) Configure the agent queue manager to run as an MSCS resource.
For information on how to do this, see [“Putting a queue manager under MSCS control” on page 615](#).

- b) Create the agent on the primary machine in the cluster.

The agent should be configured to connect to the agent queue manager using the BINDINGS transport. Ensure that you create all of the objects on the queue manager for this agent. For information on how to do this, see [Setting up the agent](#).

- c) Modify the agent to run as a Windows service, and configure it so that it does not start automatically when Windows restarts by setting the **Startup Type** field for the agent service in the Windows Services tool to Manual.

For more information, see [Starting an MFT agent as a Windows service](#).

- d) Ensure that the agent queue manager (which is under MSCS control) is running on the secondary machine.

The agent that is created on this machine will connect to the queue manager using the BINDINGS transport, and so needs to be available when the agent is created.

- e) Repeat step “2” on page 862 and step “3” on page 862 of Scenario 2 on the secondary machine.

This ensures that the file structure for logs, properties, and so on, exists on the other machine in the cluster. Note that there is no need to create the queue manager objects as in step “2” on page 862.

- f) Add the agent as a 'Generic Service' under MSCS control.

To do this:

- a. Right-click the cluster and select the **Role -> Add Resource -> 'Generic Service'**.

- b. From the list of Windows services, select the agent service and complete the configuration wizard by clicking **Next**.

- g) Modify the resource properties of the agent service to add the queue manager resource in the dependencies list.

This ensures that the queue manager resource is started before the agent is started.

- h) Take the queue manager resource offline and then bring the agent resource online. Verify if both the queue manager resource and the agent are started.

If a failover occurs, then the agent service and the agent queue manager will be started on the secondary machine.

Highly available agents in Managed File Transfer

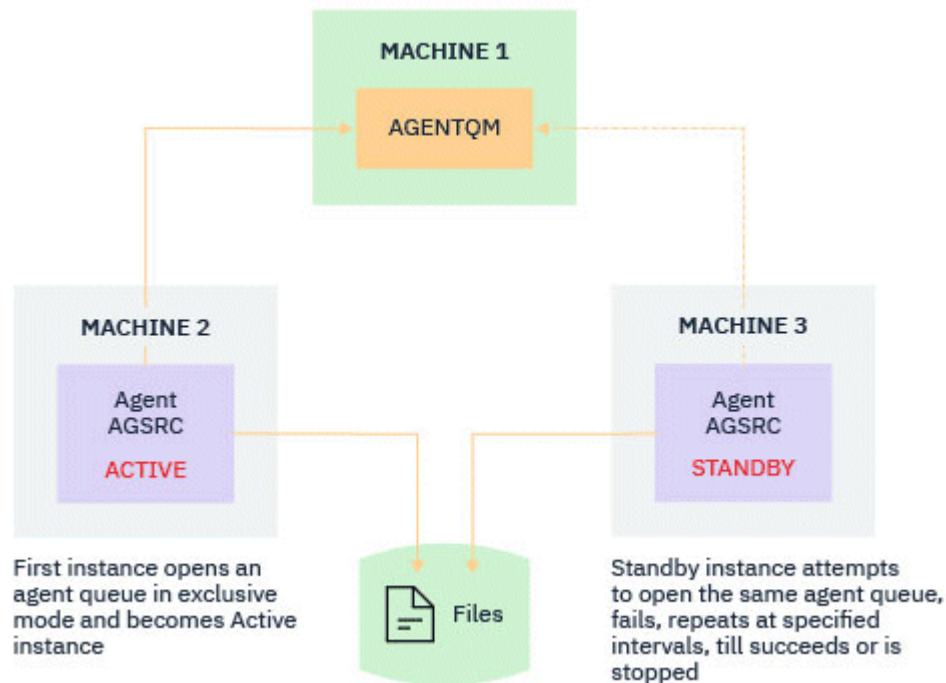
You can configure standard or bridge agents in MFT to run in a high availability (HA) configuration. A pair of agent instances with identical configurations are involved in the HA setup, where one instance is running on one machine while another instance is running on a different machine. Both instances are configured to connect to the same agent queue manager.

Overview

Only one of the two instances, called the *active instance*, is processing file transfers, while the other instance, called the *standby instance*, is in a partially initialized state and can not process any file transfers.

When an active instance fails or loses connectivity to the queue manager, the standby instance completes its initialization, becomes active and starts processing file transfers. Any inflight transfers that were in progress when the active instance failed resume from the last known check point.

The following illustration shows a common configuration of active and standby



agents:

Notes:

1. Each instance of an agent running on two different machines, with one of the instances as an *active instance*, and the other as the *standby instance*.
2. If an *active instance* fails, the *standby instance* takes over, becomes active, and starts processing file transfers.
3. The same set of agent queues is shared between both instances of the agent.
4. Both instances of the agent need access to the same shared file system in order to perform managed transfers.

The active-standby agent instance mechanism works by taking a lock on a shared resource. The agent instance that takes a lock on the shared resource becomes the active instance while the other instance (that fails to take a lock) becomes a standby instance.

The shared resource here is a new queue, `SYSTEM.FTE.HA.<agent name>`. This queue gets created automatically when an IBM MQ 9.1.4 or later agent is configured.

How the process works

To create an HA agent, you create an agent with identical configuration parameters on two machines by running the **fteCreateAgent** or **fteCreateBridgeAgent** command using the additional **-x** parameter together with the **highlyAvailable** agent property in the `agent.properties` file set to true.

Notes:

- Both configurations must point to the same agent queue manager.
- Required agent queues must be created only once on the agent queue manager.

See the **fteCreateAgent** command for more information on the **-x** parameter and the `agent.properties` file for more information on the **highlyAvailable** agent property.

Note: Running either the **fteCreateAgent** or the **fteCreateBridgeAgent** command creates an MQSC file containing the scripts that are required to create IBM MQ objects in the agent queue manager and the `SYSTEM.FTE.HA.agent name` queue. This MQSC file is created whether or not you specify the **-x** parameter.

While creating a highly available agent configuration, the **fteCreateAgent** or **fteCreateBridgeAgent** command checks the existence of an instance of the same agent present elsewhere by subscribing to the `SYSTEM.FTE/Agents/agent name` topic. If an instance of the same agent is found, then either command creates the required configuration on the file system but does not publish the creation of the agent again.

When an agent starts in HA mode:

1. The agent attempts to open the `SYSTEM.FTE.HA.agent name` queue in an exclusive GET mode.
2. If the agent opens the `SYSTEM.FTE.HA.agent name` queue successfully, this becomes the *active instance* of an agent, and a further startup process continues.
3. If the attempt to open the `SYSTEM.FTE.HA.agent name` queue in an exclusive GET mode fails with the `MQRC_OBJECT_IN_USE` reason code, it means there is already an active instance of the agent running elsewhere. Therefore, this instance becomes the *standby instance* of the agent.

The standby instance attempts to open the `SYSTEM.FTE.HA.agent name` queue at specified intervals. An additional agent property **standbyPollInterval** is provided for this purpose in the `agent.properties` file.

With the default value, the standby instance attempts to open the `SYSTEM.FTE.HA.agent name` queue every five seconds. This repeats until the instance succeeds in opening the `SYSTEM.FTE.HA.agent name` queue, or is stopped using the **fteStopAgent** command.

The **standbyPollInterval** property is also used by all instances to determine how long an instance waits between reconnection attempts if it becomes disconnected from its agent queue manager.

Multiple standby instances

All the standby instances attempt to take the `SYSTEM.FTE.HA.agent name` queue in an exclusive GET mode, and the instance that is successful, after the active instance fails, becomes the active instance.

The active instance maintains information of all known standby instances, and publishes the information as part of the agent status publication. The output of **fteShowAgentDetails** command, the agent GET REST API response, and IBM MQ Explorer MFT plug in display information about all standby instances.

See the example outputs of the **fteShowAgentDetails** command and the agent `GET` REST API response for more information.

See [MFT agent status messages](#) for examples of agent status information in XML format.

Version requirement

The active and standby agents must be IBM MQ 9.1.4 or higher.



Attention:

- You cannot configure or start IBM MQ versions prior to IBM MQ 9.1.4 in high availability modes.
- Both active and standby instances must run the same version of the code.

The version of the active and standby instances is validated to ensure that both instances are of the same version. A temporary dynamic queue is used for communication between the instances. Two agent properties, **dynamicQueuePrefix** and **modelQueueName**, defined in the `agent.properties` file, generate the name of the temporary dynamic queue.

Required information for highly available agents in Managed File Transfer

There are various types of information that you need to know about standard or bridge MFT agents that are running in a high availability configuration. This information includes the different methods by which the agent starts, how to identify the instance of the agent in the log file, and status information for the agent.

Starting an agent

An instance of an agent is running in a non-HA mode elsewhere

If an attempt is made to start another instance of the agent that is not configured as an HA agent, a check is first made to see whether a lock can be acquired on the `SYSTEM.FTE.HA.agent name` queue.

As the other instance was started in a non-HA mode, the lock on the `SYSTEM.FTE.HA.agent name` queue will be acquired by this instance. The agent continues initialization, but fails at a later point because the command queue is opened exclusively by another instance.

In this case the messages shown in the following example are logged to the `output0.log` file of the agent and the agent continues its attempt open the command queue every 30 seconds:

```
BFGMQ1045I: Agent's system queue 'SYSTEM.FTE.COMMAND.SRC' is configured as either NOSHARE or DEFSOPT(SHARED).
```

```
BFGAG0035W: The agent received MQI reason code 2042 when trying to open queue 'SYSTEM.FTE.COMMAND.SRC' on the queue manager 'MFTHAQM' with connection name 'localhost(1414)' and channel 'MFT_HA_CHN'. The agent will try the operation again every 30 seconds.
```

An instance of an agent is running in an HA mode elsewhere

If an attempt is made to start another instance of the agent that is not configured as an HA agent, a check is first made to see whether a lock can be acquired on the `SYSTEM.FTE.HA.agent name` queue.

Because the other instance has been running as an active instance, the attempt to acquire a lock fails. The instance fails to start, and the following error message is logged to the `output0.log` file of the agent:

```
BFGAG0194E: An instance of this agent is already running elsewhere.  
Hence this instance cannot continue and will end.
```

Starting the agent as a Windows service

On Windows, you can start an agent as a Windows service.

During start up, Windows starts the MFT agent in normal or HA mode. If the agent is configured to run in HA mode, the service runs as an active or standby instance, depending which instance acquires the lock first.

Identifying the instance type of an agent in the log file

Information messages are written to the `output0.log` file of the agent to indicate the type of instance. When an agent instance starts as an active instance, the following message is written:

```
BFGAG0192I: The agent has successfully initialized as an active instance.
```

When an agent instance starts as a standby instance, the following message is written:

```
BFGAG0193I: The agent has successfully initialized as a standby instance.
```

Agent status updates

As there are two instances of the same agent running, you need to have the information about both instances in the agent status publication.

Note that the active instance is the one publishing the status of both instances.

Standby instance

While publishing agent status, the active instance checks the age of the standby instance publication.

There are two additional properties in the `agent.properties` file for this purpose:

- **standbyStatusExpiry** is the expiry time for the standby status message to be put to the command queue of the agent. The message expires if the active instance of an agent does not process this message in that period.

By default, the value of **standbyStatusExpiry** is 30 seconds. The message is also a low priority, 9, message to allow priority processing of transfer requests over standby status messages.

- **standbyStatusPublishInterval** sets the frequency at which the standby instance publishes its state.

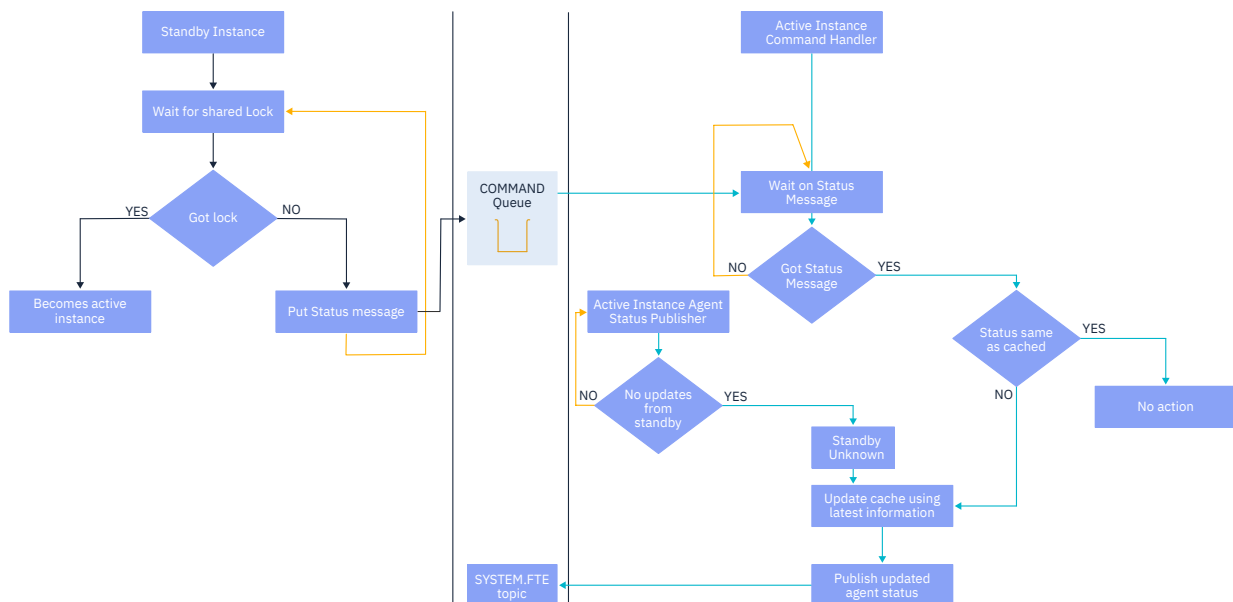
Active instance

The active instance does the following to process status updates from the standby instance:

1. Gets the message from the `SYSTEM.FTE.COMMAND.<agent name>` queue and delegates the message processing to a worker thread.
2. The worker thread retrieves the contents from the message body, updates the agent status object with standby instance information, and notifies the agent status publisher to publish the status.
3. The agent status publisher publishes the status.

Note that optimizations are done here to cache the standby status information. When a request is made, the agent status publisher checks the new status with the cached status, and publishes only if there is a difference.

The following diagram describes the flow the active or standby instances follow to publish the status of an agent:



Discarding instances, failover, and maintenance in highly available agents

Highly available Managed File Transfer instances can be discarded, can be failed in various different ways, and might need maintenance.

Discarding the standby instance status

There can be situations where the active instance is busy with transfers and is unable to process standby instance status messages, or the standby instance has failed, or is not publishing status messages for any reason.

In such scenarios the active agent that was aware of presence of standby instance waits for the value specified by the **standbyStatusDiscardTime** property in the `agent.properties` file before removing the standby instance from its list. The default value for this property is 600 seconds, which is twice that of the **standbyStatusPublishInterval** property.

Failing over an instance normally

You must use the **fteStopAgent** command with the **-i** option to carry out a normal failover.

This ensures that the active instance is stopped immediately. If you stop an agent without the **-i** option, the agent continues to run until all ongoing transfers are completed by the active instance, therefore, the failover might take a long time.

Any inflight transfers resume from the last known check point.

Failing over an instance in other situations

If an active instance ends in a way that is not normal, or the whole machine fails, the connection to the agent queue is broken, and the queue manager closes all open queues, including the `SYSTEM.FTE.HA.<agent name> queue`, and connections.

Due to this, the standby instance acquires the exclusive GET and completes the rest of the agent initialization.

Again, any inflight transfers resume from the last known check points.

If a connection to the queue manager breaks

Client mode

An agent process consists of several threads. Other than the default threads, for example, a thread that publishes agent status at regular intervals, every transfer request is handled with a set of threads that end after a transfer completes.

Many of these threads connect to the agent queue manager and put and get messages. It is possible that any of these connections can break due to a network issue or a queue manager failing. When any thread detects a connection broken problem, the thread informs the main thread to initiate recovery, and ends.

The main thread then launches another thread to wait for a connection to the queue manager being established. Once reconnected, an attempt is made to acquire the exclusive GET for the agent. If that succeeds, the agent continues to complete the recovery and becomes the active instance. If the attempt to acquire the exclusive GET fails, the instance becomes a standby.

Bindings mode

When connecting in bindings mode, if an agent loses connection, the agent process ends. The process controller handles the restarting of the agent. When an agent restarts, it goes through the process of attempting to acquire the exclusive GET for itself.

If the agent succeeds, it becomes an active instance; otherwise the agent becomes a standby instance.

Applying maintenance level upgrades

The steps for applying maintenance to highly available agents are similar to those documented for multi-instance queue managers. For more information, see [Applying maintenance level updates to multi-instance queue managers on Windows](#) or [Applying maintenance level updates to multi-instance queue managers on AIX](#), or [Applying maintenance level updates to multi-instance queue managers on Linux](#).

You must stop the agent running on the machine where the maintenance level is to be applied, before applying maintenance. If you are updating an active instance, for continuity of transfers, you must failover the active instance to a standby instance.

Once the upgrade is complete, you must start the agent instance, failover the current active instance to the upgraded instance, and then upgrade the standby instance.

Migrating agents from an earlier version of the product

Agents migrated from versions of IBM MQ prior to IBM MQ 9.1.4 run as non highly available. You can make them run in high availability mode by following the procedure in [Migrating Managed File Transfer agents from an earlier version](#).

Configuring an MFT logger

When Managed File Transfer transfers files, it publishes information about its actions to a topic on the coordination queue manager. The database logger is an optional component of Managed File Transfer that you can use to copy this information into a database for analysis and auditing purposes.

There are three versions of the logger:

-  stand-alone file logger
- stand-alone database logger
- Java Platform, Enterprise Edition (Java EE) logger

Loggers on IBM i



Managed File Transfer loggers are not supported on the IBM i platform.

Stand-alone file logger



The stand-alone file logger is a Java process that runs on two types of systems:

- A system that hosts the coordination queue manager.
- A system that hosts a queue manager with connectivity to the coordination queue manager.

The stand-alone file logger uses IBM MQ bindings to connect to its associated logger queue manager. If the logger is configured to use the FILE mode, it can also connect to the logger queue manager in client mode. The stand-alone logger is created by using the **fteCreateLogger** command. The Managed File Transfer logger has a set of configuration properties that you can specify in the `logger.properties` file.

For example, you can set the following properties to specify that the stand-alone logger uses a client connection to the logger queue manager.

- `wmqfte.queue.manager.host`
- `wmqfte.queue.manager.port`
- `wmqfte.queue.manager.channel`

For more information, see [The MFT logger.properties file](#).

 You can run the stand-alone file logger as a Windows service to help ensure that the file logger continues running when you log off from your Windows session, and it can be configured to start automatically when a system restarts. For more information, see [“Installing the MFT stand-alone file logger” on page 869](#).

The stand-alone file logger is not supported on the following platforms:

-  z/OS
-  IBM i

Stand-alone database logger

The stand-alone database logger is a Java application that you install on a system that hosts a queue manager and a database. The stand-alone database logger is often installed on the same system as the coordination queue manager, however it can also be installed on the same system as any queue manager which has connectivity to the coordination queue manager. The stand-alone database logger uses IBM MQ bindings to connect to its associated queue manager, and a type 2 or type 4 JDBC driver to connect to a Db2 or Oracle database. These types of connection are required because the stand-alone database

logger uses the queue manager's XA support to coordinate a global transaction over both the queue manager and database, protecting the data.

 **Windows** If you are using a Windows system, you can run the stand-alone loggers as Windows services to ensure that the loggers continue running when you log off from your Windows session. For more information, see [“Installing the MFT stand-alone database logger” on page 885](#) for a stand-alone database logger.

Java EE database logger

The Java EE database logger is provided as an EAR file, which you install into an application server. This can be more convenient than using the stand-alone database logger if you have an existing Java EE application server environment available because the Java EE database logger can be managed alongside your other enterprise applications. You can also install the Java EE database logger on a separate system to the systems hosting your IBM MQ server and database. The Java EE database logger is supported for use with Db2 and Oracle databases. The Java EE database logger also supports Oracle Real Application Clusters when installed on WebSphere Application Server 7.0.

For instructions on how to configure a logger, see the following topics:

- [“Installing the MFT stand-alone file logger” on page 869](#)
- [“Installing the MFT stand-alone database logger” on page 885](#)
- [“Installing the Java EE database logger for MFT” on page 891](#)

Related tasks

[“Using MFT with a remote database” on page 887](#)

You can use the Managed File Transfer logger to communicate with a database on a remote system.

Related reference

[MFT logger error handling and message rejection](#)

[MFT logger configuration properties](#)

Installing the MFT stand-alone file logger

The Managed File Transfer (MFT) stand-alone file logger is a Java process that can connect to a coordination queue manager by using IBM MQ bindings mode or client mode. To define a stand-alone file logger, use the **fteCreateLogger** command and follow the steps in this topic.

About this task

Note: The MFT stand-alone file logger cannot be configured and executed on a redistributable MFT agent installation image.

For more information about the stand-alone file logger, see [“Configuring an MFT logger” on page 868](#). The steps in this topic configure a logger to connect to a coordination queue manager. For alternative logger configurations see [“Alternative configurations for an MFT stand-alone logger” on page 890](#).

The stand-alone file logger is not supported on the following platforms:

-  **z/OS** z/OS
-  **IBM i** IBM i

Procedure

1. Ensure that you have the Managed File Transfer Logger component installed. For more information, see [Managed File Transfer product options](#).
2. Run the **fteCreateLogger** command specifying the coordination queue manager, and setting the parameter **-loggerType** to FILE to create your stand-alone file logger.

For more information, see [fteCreateLogger](#).

- Optional: If you want to use a custom format then you can modify the XML file created by the **fteCreateLogger** command. The log format definition is located in the `FileLoggerFormat.xml` file.

For more information, see “MFT stand-alone file logger format” on page 871.

- Run the MQSC commands, provided by the **fteCreateLogger** command, against your coordination queue manager to create the logger queues.

- Identify a user to run the logger process and configure permissions for that user.

For more information, see “Configuring user access for an MFT stand-alone file logger” on page 885.

- Optional: Configure the stand-alone file logger further by editing the `logger.properties` file created when you ran the **fteCreateLogger** command.

This file is a Java properties file that consists of key-value pairs. The `logger.properties` file is in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory. For more information about available properties and their affects, see [MFT logger configuration properties](#).

-  **Windows**

Optional: If you are using a Windows system, run the stand-alone file logger as a Windows service.

Run the **fteModifyLogger** command with the **-s** parameter. For more information, see [fteModifyLogger](#).

- Start the stand-alone file logger with the **fteStartLogger** command.

For more information, see [fteStartLogger](#).

If you carried out the previous step and used the **fteModifyLogger** command with the **-s** parameter on Windows, the stand-alone file logger starts as a Windows service.

- Check the logger output. The stand-alone file logger generates two types of output, file transfer audit data and logger diagnostic data.

The file transfer audit data can be found in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/loggers/logger_name/logs`. The logger diagnostic data can be found in `MQ_DATA_PATH/mqft/logs/coordination_qmgr_name/loggers/logger_name`.

- Stop the logger by using the **fteStopLogger** command.

For more information, see [fteStopLogger](#).

Results

Related tasks

[“Configuring user access for an MFT stand-alone file logger” on page 885](#)

In a test environment, you can add any new privileges needed to your normal user account. In a production environment, you are recommended to create a new user with the minimum permissions required to do the job.

Related reference

[MFT logger configuration properties](#)

[fteStartLogger](#) (start an MFT logger)

[fteCreateLogger](#) (create an MFT file or database logger)

[fteModifyLogger](#) (run an MFT logger as a Windows service)

[fteStopLogger](#) (stop an MFT logger)

[“MFT stand-alone file logger format” on page 871](#)

The format of message information written by the file logger can be defined in the `FileLoggerFormat.xml` file.

[Authorities for the MFT logger](#)

The format of message information written by the file logger can be defined in the `FileLoggerFormat.xml` file.

The configuration directory for the logger is located in `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name`. When creating a new file logger, a version of this file is created that contains a default set of definitions used by the file logger. For more information about the default log format definition, see [MFT stand-alone file logger default log format](#).

If you want to specify your own custom log format, edit the `FileLoggerFormat.xml` file.

A custom log format definition

A log format definition consists of a set of message types with each message type having a format definition. A format definition for a message type consists of a set of inserts provided in XPATH format and a separator that is used to separate each insert. The ordering of the inserts determines the order in which the content is placed in the lines generated for output to the log files. For example, this is the definition for the `callStarted` message type:

```
<callStarted>
  <format>
    <inserts>
      <insert type="user"    width="19" ignoreNull="false">/transaction/action/
        @time</insert>
      <insert type="user"    width="48" ignoreNull="false">/transaction/@ID</insert>
      <insert type="system"  width="6"  ignoreNull="false">type</insert>
      <insert type="user"    width="0"  ignoreNull="false">/transaction/agent/
        @agent</insert>
      <insert type="user"    width="0"  ignoreNull="false">/transaction/agent/@QMgr</insert>
      <insert type="user"    width="0"  ignoreNull="false">/transaction/job/name</insert>
      <insert type="user"    width="0"  ignoreNull="true">/transaction/transferSet/
        call/command/@type</insert>
      <insert type="user"    width="0"  ignoreNull="true">/transaction/transferSet/
        call/command/@name</insert>
      <insert type="system"  width="0"  ignoreNull="true">callArguments</insert>
    </inserts>
    <separator></separator>
  </format>
</callStarted>
```

This format produces a line in the log file like this:

```
2011-11-25T10:53:04;414d5120514d5f67627468696e6b20206466cf4e20004f02;[CSTR];
AGENT1;AGENT_QM;Managed Call;executable;echo;call test;
```

The inserts provided in the format definition are in the order in which the information appears on the line in the log file. For more information on the XML schema defining the format for the `FileLoggerFormat.xml` file, see [Stand-alone file logger format XSD](#).

Message types

The FTE agents write a range of different message types to the `SYSTEM.FTE/Log` sub-topic. For more information, see [SYSTEM.FTE topic](#). The log file definition can contain format definitions for these types of messages:

```
callCompleted
callStarted
monitorAction
monitorCreate
monitorFired
notAuthorized
scheduleDelete
scheduleExpire
scheduleSkipped
scheduleSubmitInfo
scheduleSubmitTransfer
scheduleSubmitTransferSet
transferStarted
transferCancelled
```

```
transferComplete
transferDelete
transferProgress
```

For more information about the attributes that an MFT file logger can log for each message type, see “Attributes that an MFT file logger can log for each message type” on [page 874](#). The attributes can be used as XPATH inserts to customize the output of an MFT file logger.

The format of the messages can vary. The majority of message types write a single line in the log file for each log message consumed from the SYSTEM.FTE/Log sub-topic. This leads to the simple case where the XPATH addresses provided in the log format definition relate to the root of the message. These are the message types that use this method to write output:

```
callCompleted
callStarted
monitorAction
monitorCreate
monitorFired
notAuthorized
scheduleDelete
scheduleExpire
scheduleSkipped
scheduleSubmitInfo
scheduleSubmitTransfer
transferStarted
transferCancelled
transferComplete
transferDelete
```

The other method used to write a log message uses multiple lines to represent the items in a transfer set within a log message. In this case the format provided is applied to each item in the transfer set within the log message. If you want to include information that is specific to each item within the transfer set, then the XPATH provided is required to use the item as its XPATH root. These are the message types that use this method to write output:

```
scheduleSubmitTransferSet
transferProgress
```

A line of output is written for each item in the transfer set. Information that you want to be fixed for all items in a transfer set can still use XPATH addresses relative to the root of the log message. In the following simplified transferProgress format definition example it's the timestamp and transfer ID that are fixed. Any information that is relative to an item as its root will vary for each line written. In this example the source and destination file information for each item are written.

```
<transferProgress>
  <format>
    <inserts>
      <insert type="user" width="19" ignoreNull="false">/transaction/action/
        @time</insert>
      <insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
      <insert type="system" width="6" ignoreNull="false">type</insert>
      <insert type="user" width="3" ignoreNull="true">status/@resultCode</insert>
      <insert type="user" width="0" ignoreNull="false">source/file |
        source/queue</insert>
      <insert type="user" width="0" ignoreNull="false">source/file/@size |
        source/queue/@size</insert>
      <insert type="user" width="5" ignoreNull="true">source/@type</insert>
      <insert type="user" width="6" ignoreNull="true">source/@disposition</insert>
      <insert type="user" width="0" ignoreNull="false">destination/file |
        destination/queue</insert>
      <insert type="user" width="0" ignoreNull="false">destination/file/@size |
        destination/queue/@size</insert>
      <insert type="user" width="5" ignoreNull="true">destination/@type</insert>
      <insert type="user" width="9" ignoreNull="true">destination/@exist</insert>
      <insert type="user" width="0" ignoreNull="true">status/supplement</insert>
    </inserts>
    <separator></separator>
  </format>
</transferProgress>
```

This produces a log file entry of one or more lines in this format:

```
2011-11-25T13:45:16;414d5120514d5f67627468696e6b20206466cf4e20033702;[TPR0];0
;/src/test1.file;3575;file;leave ;/dest/test1.file;3575;file;overwrite;;
2011-11-25T13:45:16;414d5120514d5f67627468696e6b20206466cf4e20033702;[TPR0];0
;/src/test2.file;3575;file;leave ;/dest/test2.file;3575;file;overwrite;;
```

Insert Format

There are two types of insert available when defining a format for a message type: user and system. The type of an insert is defined in the type attribute of the insert element. Both types of inserts can also have their layout customized using the **width** and **ignoreNull** attributes of the insert element. For example:

```
<insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
```

In this example, the insert takes the information found in the log message at `/transaction/@ID` and trims or pads it to 48 characters before writing it to the log. If the content of `/transaction/@ID` is null it writes the string null after padding it to 48 characters because the `ignoreNull` attribute is set to false. If `ignoreNull` is set to true the empty string, padded to 48 characters, is written instead. Setting `width="0"` means the column width is not trimmed, it does not mean that the width is trimmed to 0. The `ignoreNull` attribute can be used in this way to detect in the log when a null is found when it was not expected. This can be useful when debugging a new log file definition.

User defined inserts

A user insert contains an XPATH address for the information to be written in that insert. This address refers to a piece of information found in the FTE log message. For more information about log message formats, see:

- [File transfer log message formats](#)
- [Scheduled file transfer log message formats](#)
- [MFT monitor log message format](#)

System defined inserts

System defined inserts contain a keyword that refers to a piece of information that either cannot be found in the log message or is not easy to define using the XPATH language.

Supported systems inserts are:

- `type` - Writes the type of the log message in a short format.
- `callArguments` - Writes the set of arguments supplied to a managed call in a space separated format.
- `transferMetaData` - Writes the set of metadata entries defined for a transfer in a comma separated `key=value` format.

The following table lists the value of "type" for system defined inserts for each message type.

Table 51. Summary of supported message types and their "type" system inserts.	
Message type	Value of "type" system insert
callCompleted	[CCOM]
callStarted	[CSTR]
monitorAction	[MACT]
monitorCreate	[MCRT]

Table 51. Summary of supported message types and their "type" system inserts. (continued)

Message type	Value of "type" system insert
monitorFired	[MFIR]
notAuthorized	[AUTH]
scheduleDelete	[SDEL]
scheduleExpire	[SEXP]
scheduleSkipped	[SSKP]
scheduleSubmitInfo	[SSIN]
scheduleSubmitTransfer	[SSTR]
scheduleSubmitTransferSet	[SSTS]
transferStarted	[TSTR]
transferCancelled	[TCAN]
transferComplete	[TCOM]
transferDelete	[TDEL]
transferProgress	[TPRO]

Related reference

[MFT stand-alone file logger default log format](#)

[Stand-alone file logger format XSD](#)

[SYSTEM.FTE topic](#)

[File transfer log message formats](#)

[Scheduled file transfer log message formats](#)

[MFT monitor log message format](#)

Attributes that an MFT file logger can log for each message type

You can use the attributes that a Managed File Transfer (MFT) file logger can log for each message type as XPATH inserts to customize the output of an MFT file logger.

Message type: callCompleted

Table 52. Attributes for callCompleted

Attribute name	Description
/transaction/action/@time	Date and time when this event was generated.
/transaction/@ID	Managed call identifier.
/transaction/status/@resultCode	Result of the managed call.
type	The type of log message in a short format. For this message type, the value is CCOM.
/transaction/agent/@agent	Agent to which the managed call was submitted.

Table 52. Attributes for callCompleted (continued)

Attribute name	Description
/transaction/agent/@QMgr	Agent queue manager.
/transaction/job/name	Job name
/transaction/transferSet/call/command/@type	Command type (executable, antscript or jcl).
/transaction/transferSet/call/command/@name	Name of command executed by the managed call.
callArguments	Arguments passed to the command.
/transaction/transferSet/call/callResult/@outcome	Outcome of command execution – success, failure or error.
/transaction/transferSet/call/callResult/result/error	Detailed information on the error.

Message type: callStarted

Table 53. Attributes for callStarted

Attribute name	Description
/transaction/action/@time	Date and time when this event was generated.
/transaction/@ID	Managed call identifier.
type	The type of log message in a short format. For this message type, the value is CSTR.
/transaction/agent/@agent	Agent to which the managed call was submitted.
/transaction/agent/@QMgr	Agent queue manager.
/transaction/job/name	Job name.
/transaction/transferSet/call/command/@type	Command type (executable, antscript or jcl).
/transaction/transferSet/call/command/@name	Name of command to be executed.
callArguments	List of arguments passed to the command.

Message type: monitorAction

Table 54. Attributes for monitorAction

Attribute name	Description
/monitorLog/action/@time	Date and time when this event was generated.
/monitorLog/@referenceId	Resource monitor identifier.
type	The type of log message in a short format. For this message type, the value is MACT.
/monitorLog/status/@resultCode	Result of resource monitor action.
/monitorLog/@monitorName	Name of the resource monitor.
/monitorLog/monitorAgent/@agent	Name of the agent where the resource monitor is running.
/monitorLog/monitorAgent/@QMgr	Agent queue manager.

Table 54. Attributes for monitorAction (continued)	
Attribute name	Description
/monitorLog/action	Type of resource monitor action – create, delete, start, stop, triggerSatisfied, triggerNotSatisfied, triggerFail.

Message type: monitorCreate

Table 55. Attributes for monitorCreate	
Attribute name	Description
/monitorLog/action/@time	Date and time when this event was generated.
/monitorLog/@referenceId	Resource monitor identifier.
type	The type of log message in a short format. For this message type, the value is MCRT.
/monitorLog/@monitorName	Name of the resource monitor.
/monitorLog/monitorAgent/@agent	Name of the agent where the resource monitor will be created.
/monitorLog/monitorAgent/@QMgr	Name of the agent queue manager.
/monitorLog/action	Resource monitor create action.

Message type: monitorFired

Table 56. Attributes for monitorFired	
Attribute name	Description
/monitorLog/action/@time	Date and time when this event was generated.
/monitorLog/@referenceId	Resource monitor identifier.
type	The type of log message in a short format. For this message type, the value is MFIR.
/monitorLog/status/@resultCode	Result of the resource monitor fired action.
/monitorLog/@monitorName	Name of the resource monitor.
/monitorLog/monitorAgent/@agent	Name of the agent where the resource monitor is running.
/monitorLog/monitorAgent/@QMgr	Name of the agent queue manager.
/monitorLog/action	Resource monitor fired.
/monitorLog/references/taskRequest	Details of a transfer task request submitted by this resource monitor.

Message type: notAuthorized

Table 57. Attributes for notAuthorized	
Attribute name	Description
/notAuthorized/action/@time	Date and time when this event was generated.

Table 57. Attributes for notAuthorized (continued)

Attribute name	Description
/notAuthorized/@ID	Identifier for the event
type	The type of log message in a short format. For this message type, the value is AUTH.
/notAuthorized/status/@resultCode	Result code provided in the event.
/notAuthorized/action	Not authorized.
/notAuthorized/authority	Authority required to perform an action.
/notAuthorized/originator/userID	Id of the user that submitted a request.
/notAuthorized/originator/mqmdUserID	MQ User id.
/notAuthorized/status/supplement	Additional information on the not authorized event.

Message type: scheduleDelete

Table 58. Attributes for scheduleDelete

Attribute name	Description
/schedulelog/action/@time	Date and time when this event was generated.
/schedulelog/@ID	Identifier for the scheduled transfer.
type	The type of log message in a short format. For this message type, the value is SDEL.
/schedulelog/status/@resultCode	Result of scheduled transfer delete action.
/schedulelog/sourceAgent/@agent	Name of the agent to which the action was submitted.
/schedulelog/action	Delete.
/schedulelog/originator/userID	User that submitted the scheduled transfer delete request.
/schedulelog/status/supplement	Additional information on the outcome of scheduled transfer delete request

Message type: scheduleExpire

Table 59. Attributes for scheduleExpire

Attribute name	Description
/schedulelog/action/@time	Date and time when this event was generated.
/schedulelog/@ID	Identifier for the scheduled transfer.
type	The type of log message in a short format. For this message type, the value is SEXP.
/schedulelog/status/@resultCode	Result code of scheduled transfer expire action.
/schedulelog/sourceAgent/@agent	Name of the agent to which the action was submitted.
/schedulelog/action	Expire

Table 59. Attributes for scheduleExpire (continued)	
Attribute name	Description
/schedulelog/originator/userID	User Id that created the scheduled transfer.
/schedulelog/status/supplement	Additional information on the outcome of scheduled transfer expire request.

Message type: scheduleSkipped

Table 60. Attributes for scheduleSkipped	
Attribute name	Description
/schedulelog/action/@time	Date and time when this event was generated.
/schedulelog/@ID	Identifier for the scheduled transfer.
type	The type of log message in a short format. For this message type, the value is SSKP.
/schedulelog/status/@resultCode	Result of scheduled transfer skip action.
/schedulelog/sourceAgent/@agent	Name of the agent to which the action was submitted.
/schedulelog/action	Schedule Skipped
/schedulelog/originator/userID	User Id that created the scheduled transfer.
/schedulelog/status/supplement	Additional information on the outcome of scheduled transfer skipped request.

Message type: scheduleSubmitInfo

Table 61. Attributes for scheduleSubmitInfo	
Attribute name	Description
/schedulelog/action/@time	Date and time when this event was generated.
/schedulelog/@ID	Identifier for the scheduled transfer.
type	The type of log message in a short format. For this message type, the value is SSIN.
/schedulelog/status/@resultCode	Result of scheduled transfer submit action.
/schedulelog/sourceAgent/@agent	Name of the agent to which the action was submitted.
/schedulelog/action	Transfer request submit.
/schedulelog/originator/userID	User Id that created the scheduled transfer.
/schedulelog/schedule/submit/@timezone	Time zone used for submission of transfer request.
/schedulelog/schedule/repeat/frequency/@interval	Intervals at which the scheduled transfer occurs.
/schedulelog/schedule/repeat/expireCount	The number times the scheduled transfer occurs.
/schedulelog/status/supplement	Additional information on the outcome of scheduled transfer skipped request.

Message type: scheduleSubmitTransfer

Table 62. Attributes for scheduleSubmitTransfer	
Attribute name	Description
/schedulelog/action/@time	Date and time when this event was generated.
/schedulelog/@ID	Identifier for the scheduled transfer.
type	The type of log message in a short format. For this message type, the value is SSTR.
/schedulelog/sourceAgent/@agent	Name of the source agent for the transfer request.
/schedulelog/sourceAgent/@QMgr	Name of the source agent queue manager.
/schedulelog/destinationAgent/@agent	Name of the destination agent for the transfer request.
/schedulelog/destinationAgent/@QMgr	Name of the destination agent queue manager.

Message type: scheduleSubmitTransferSet

Table 63. Attributes for scheduleSubmitTransferSet	
Attribute name	Description
/schedulelog/action/@time	Date and time when this event was generated.
/schedulelog/@ID	Identifier for the scheduled transfer.
type	The type of log message in a short format. For this message type, the value is SSTs.
source/file	Name of the source file in the transfer request.
source/queue	Name of the queue in the transfer request. Present in the source type is Queue.
source/@type	Source type
source/@disposition	Delete or retain source file after transfer is completed.
destination/file	Name of the destination file.
destination/queue	Name of the destination queue. Present in the destination type is Queue.
destination/@type	Destination type
destination/@exist	Action to take if destination file already exists. Overwrite or Error.

Message type: transferStarted

Table 64. Attributes for transferStarted	
Attribute name	Description
/transaction/action/@time	Date and time when this event was generated.
/transaction/@ID	Identifier of the transfer.

<i>Table 64. Attributes for transferStarted (continued)</i>	
Attribute name	Description
type	The type of log message in a short format. For this message type, the value is TSTR.
/transaction/status/@resultCode	Result code.
/transaction/sourceAgent/@agent	Name of the source agent.
/transaction/sourceAgent/@QMgr	Name of the source agent queue manager.
/transaction/sourceAgent/@agentType	Agent type – Standard or Bridge.
/transaction/destinationAgent/@agent	Name of the destination agent.
/transaction/destinationAgent/@QMgr	Name of the destination agent queue manager.
/transaction/originator/userID	User id that submitted transfer request.
/transaction/job/name	Job name
/transaction/scheduleLog/@ID	Identifier of that schedule transfer that triggered this transfer request.
transferMetaData	Metadata associated with transfer request.

Message type: transferCancelled

<i>Table 65. Attributes for transferCancelled</i>	
Attribute name	Description
/transaction/action/@time	Date and time when this event was generated.
/transaction/@ID	Identifier of the transfer.
type	The type of log message in a short format. For this message type, the value is TCAN.
/transaction/status/@resultCode	Result code.
/transaction/sourceAgent/@agent	Name of the source agent.
/transaction/sourceAgent/@QMgr	Name of the source agent queue manager.
/transaction/sourceAgent/@agentType	Agent type – Standard or Bridge.
/transaction/destinationAgent/@agent	Name of the destination agent.
/transaction/destinationAgent/@QMgr	Name of the destination agent queue manager.
/transaction/destinationAgent/@agentType	Destination agent type – Standard or Bridge.
/transaction/originator/userID	User id that submitted transfer request.
/transaction/job/name	Job name.
/transaction/status/supplement	Additional information on the event.

Message type: transferComplete

<i>Table 66. Attributes for transferComplete</i>	
Attribute name	Description
/transaction/action/@time	Date and time when this event was generated.

Table 66. Attributes for transferComplete (continued)

Attribute name	Description
/transaction/@ID	Identifier of the transfer.
type	The type of log message in a short format. For this message type, the value is TCOM.
/transaction/status/@resultCode	Result code.
/transaction/sourceAgent/@agent	Name of the source agent.
/transaction/sourceAgent/@QMgr	Name of the source agent queue manager.
/transaction/sourceAgent/@agentType	Source agent type.
/transaction/destinationAgent/@agent	Name of the destination agent.
/transaction/destinationAgent/@QMgr	Name of the destination agent queue manager.
/transaction/destinationAgent/@agentType	Destination agent type.
/transaction/originator/userID	User Id that submitted transfer request.
/transaction/job/name	Job name.
/transaction/status/supplement	Additional information on transfer completion.
transferMetaData	Metadata associated with transfer request.

Message type: transferDelete

Table 67. Attributes for transferDelete

Attribute name	Description
/transaction/action/@time	Date and time when this event was generated.
/transaction/@ID	Identifier of the transfer.
type	The type of log message in a short format. For this message type, the value is TDEL.
/transaction/status/@resultCode	Result code.
/transaction/sourceAgent/@agent	Name of the source agent.
/transaction/sourceAgent/@QMgr	Name of the source agent queue manager.
/transaction/sourceAgent/@agentType	Source Agent type – Standard or Bridge.
/transaction/destinationAgent/@agent	Name of the destination agent.
/transaction/destinationAgent/@QMgr	Name of the destination agent queue manager.
/transaction/destinationAgent/@agentType	Destination agent type – Standard or Bridge.
/transaction/originator/userID	User Id that submitted transfer request.
/transaction/job/name	Job name.
/transaction/status/supplement	Additional information of the event.

Message type: transferProgress

Table 68. Attributes for transferProgress	
Attribute name	Description
/transaction/action/@time	Date and time when this event was generated.
/transaction/@ID	Identifier of the transfer.
type	The type of log message in a short format. For this message type, the value is TPRO.
status/@resultCode	Result code for the transfer of this item.
source/file	Name of the source file.
source/queue	Name of the source queue.
source/file/@size	Size of the source file.
source/queue/@size	Size of the message. Set as “null” always.
source/@type	Source type
source/@disposition	Action to perform after transfer is completed – delete or leave the source file.
source/file/@alias	The Connect:Direct transfer name, when transferring from a Connect:Direct node to a Managed File Transfer agent using the Connect:Direct bridge.
source/queue/@alias	Alias queue name if source type is queue.
source/@correlationBoolean1	A flag indicating whether a transfer from a Connect:Direct node to a Managed File Transfer agent using the Connect:Direct bridge calls a user-defined Connect:Direct process.
source/@correlationNum1	The Connect:Direct process identifier, when transferring from a Connect:Direct node to a Managed File Transfer agent using the Connect:Direct bridge.
source/@correlationString1	The name of the Connect:Direct process, when transferring from a Connect:Direct node to a Managed File Transfer agent using the Connect:Direct bridge.
destination/file	Name of the destination file.
destination/queue	Name of the destination queue.
destination/file/@size	Destination file size.
destination/queue/@size	Destination message size – Set as “null” always.
destination/@type	Destination type.
destination/@exist	Action to perform when destination exists – Overwrite or Error.
destination/file/@alias	The Connect:Direct transfer name, when transferring from a Managed File Transfer agent to a Connect:Direct node using the Connect:Direct bridge.

Table 68. Attributes for transferProgress (continued)

Attribute name	Description
destination/queue/@alias	Alias queue name for destination. Valid only if destination type is queue.
destination/file/@truncateRecords	Truncate destination records longer than LRECL dataset attributes. Valid only if destination type is dataset.
destination/@correlationBoolean1	A flag indicating whether a transfer from a Managed File Transfer agent to a Connect:Direct node using the Connect:Direct bridge calls a user-defined Connect:Direct process.
destination/@correlationNum1	The Connect:Direct process identifier, when transferring from a Managed File Transfer agent to a Connect:Direct node using the Connect:Direct bridge.
destination/@correlationString1	The name of the Connect:Direct process, when transferring from a Managed File Transfer agent to a Connect:Direct node using the Connect:Direct bridge.
status/supplement	Additional information on the event.

Excluding message types from the MFT stand-alone file logger

If you want to exclude a certain message type from the file logger output, you can use empty message type elements.

Example

For example, the following format definition stops transferProgress messages being output by the file logger.

```
<?xml version="1.0" encoding="UTF-8"?>
<logFormatDefinition xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="1.00"
  xsi:noNamespaceSchemaLocation="FileLoggerFormat.xsd">
  <messageTypes>
    <transferProgress></transferProgress>
  </messageTypes>
</logFormatDefinition>
```

Defining custom formats for the MFT stand-alone file logger

It is possible to define a subset of custom message types within a log format definition to reduce the amount of configuration required to customize your log file format.

About this task

If a messageTypes element is not included in the FileLoggerFormat.xml file, the format for that message type uses the default format. You need only to specify the formats you want to be different from the default.

Example

In this example, the format definition replaces the default format for the `transferStarted` message type, with this reduced version that outputs only the user that started the transfer. All other message types use the default format because they are not included in this log format definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<logFormatDefinition xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" version="1.00"
  xsi:noNamespaceSchemaLocation="FileLoggerFormat.xsd">
  <messageTypes>
    <transferStarted>
      <format>
        <inserts>
          <insert type="user" width="19" ignoreNull="false"/>transaction/action/
            @time</insert>
          <insert type="user" width="48" ignoreNull="false"/>transaction/@ID</insert>
          <insert type="system" width="6" ignoreNull="false">type</insert>
          <insert type="user" width="0" ignoreNull="true">/transaction/originator/
            userID</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </transferStarted>
  </messageTypes>
</logFormatDefinition>
```

Related reference

[MFT stand-alone file logger default log format](#)

[Stand-alone file logger format XSD](#)

Reducing duplicate messages in the MFT stand-alone file logger

Duplicate log messages can occur in the log of the stand-alone file logger. By using the `logger.properties` file you can tune the stand-alone file logger and reduce the number of duplicates.

Duplicate messages in the file logger log

In the case of a failure, a log message might be written to the log of the stand-alone file logger without the consumption of the log message from the `SYSTEM.FTE/Log#` topic being committed to IBM MQ. If this happens, when the stand-alone file logger restarts it will retrieve the same message a second time and write it to the log file again. Plan to handle the possibility of these duplicates when looking at the log files either manually or when processing them automatically. To assist in the detection of duplicates, the stand-alone file logger outputs the following message to the log file when it starts:

```
BFGDB0054I: The file logger has successfully started
```

Duplicates always happen around the start time of the stand-alone file logger, because this is when the last message read before the previous instance failed is processed. By knowing when the new instance has started you can detect if duplicates should be expected, and whether they need to be handled or not.

Reducing the number of duplicates

The stand-alone file logger groups together log messages it processes into transactions to improve performance. This batch size is the maximum number of duplicate messages you may see in the case of a failure. To reduce the number of duplicates you can tune the following property in the `logger.properties` file:

```
wmqfte.max.transaction.messages
```

For example, by setting this to 1 the maximum number of duplicated messages is reduced to 1. Be aware that modifying this value has an effect on the performance of your stand-alone file logger so thorough testing is required to ensure this does not adversely affect your system.

The `logger.properties` file is in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory. For more information on available properties and their effects, see [MFT logger configuration properties](#)

Configuring user access for an MFT stand-alone file logger

In a test environment, you can add any new privileges needed to your normal user account. In a production environment, you are recommended to create a new user with the minimum permissions required to do the job.

About this task

You must install the stand-alone file logger and IBM MQ on a single system. Configure the user's permissions as follows:

Procedure

1. Ensure that the user has permission to read and, where necessary, execute, the files installed as part of the Managed File Transfer installation.
2. Ensure that the user has permission to create and write to any file in the `logs` directory which is in the configuration directory. This directory is used for an event log, and if necessary for diagnostic trace and First Failure Data Capture (FFDC) files.
3. Ensure that the user has its own group, and is also not in any groups with wide-ranging permissions on the coordination queue manager. The user should not be in the `mqm` group. On certain platforms, the staff group is automatically given queue manager access as well; the stand-alone file logger user should not be in the staff group. You can view authority records for the queue manager itself and for objects in it by using the IBM MQ Explorer. Right-click the object and select **Object Authorities > Manage Authority Records**. At the command line, you can use the commands [dspmqaut](#) (display authority) or [dmpmqaut](#) (dump authority).
4. Use the **Manage Authority Records** window in the IBM MQ Explorer or the [setmqaut](#) (grant or revoke authority) command to add authorities for the user's own group (on AIX, IBM MQ authorities are associated with groups only, not individual users). The authorities required are as follows:
 - Connect and Inquire on the queue manager (the IBM MQ Java libraries require Inquire permission to operate).
 - Subscribe permission on the `SYSTEM.FTE.LOG.RJCT.logger_name` queue.
 - Put permission on the `SYSTEM.FTE.LOG.RJCT.logger_name` queue.
 - Get permission on the `SYSTEM.FTE.LOG.CMD.logger_name` queue.

The reject and command queue names given are the default names. If you chose different queue names when you configured the stand-alone file logger queues, add the permissions to those queue names instead.

Installing the MFT stand-alone database logger

Complete these steps to install and configure the stand-alone database logger.

About this task

Important: Managed File Transfer loggers are not supported on the IBM i platform.

For more information about the stand-alone database logger, see [“Configuring an MFT logger”](#) on page 868.

Note: You cannot run more than one database logger (stand-alone or Java EE) against the same schema in a database at any one time. Attempting to do so would result in clashes when attempting to write transfer log data to the database.

Procedure

1. Install your database software using the documentation for your database.
If JDBC support is an optional component for your database, you must install this component.
2. Run the **fteCreateLogger** command setting the parameter **-loggerType** to DATABASE to create your stand-alone database logger. For more information, see [fteCreateLogger](#).

The default schema name is FTELOG. If you use a schema name other than FTELOG, you must edit the provided SQL file appropriate to your database, `ftelog_tables_db2.sql` or `ftelog_tables_oracle.sql`, to reflect this schema name before proceeding to the next step. For more information, see `wmqfte.database.schema` in [MFT logger configuration properties](#).

3. Create the required database tables using your database's tools.

Multi On Multiplatforms, the files `ftelog_tables_db2.sql` and `ftelog_tables_oracle.sql` contain SQL commands that you can run to create the tables.

z/OS On z/OS, the file that you need to run depends on the version of Db2 for z/OS that you are using:

- For Db2 for z/OS 9.0 and earlier, run the file `ftelog_tables_zos.sql` to create the tables. This file creates the tables using an INTEGER data type for fields which denote the sizes of files that are transferred and the table ID associated with each transfer.
 - For Db2 for z/OS 9.1 and later, run the file `ftelog_tables_zos_bigint.sql` to create the tables. This file creates the tables using a BIGINT data type for fields which denote the sizes of files that are transferred and the table ID associated with each transfer.
4. If you are using an Oracle database, add `USE_SID_AS_SERVICE_listener_name=on` to your `listener.ora` file, where `listener_name` is the name of the listener on your database server. Then restart the listener.

This parameter specifies that the database connection treats the Oracle system identifier (SID) as the service name for the database. This parameter is required because the MFT database logger connects to Oracle databases only by using a SID but later Oracle databases (from version 12c) use a service name. For more information, see the documentation for your version of Oracle, for example [Oracle Net Listener Parameters in the listener.ora File](#). If you do not modify the `listener.ora` file, you might see the following message when you run the logger:

```
ORA-12505, TNS:listener does not currently know of SID given in connect descriptor
```

5. Run the MQSC commands, provided by the **fteCreateLogger** command, against your logger command queue manager to create the logger queues. The stand-alone database logger uses two queues on the coordination queue manager. The first queue is a command queue where messages to control the operation of the stand-alone database logger are placed. The default name of this command queue is `SYSTEM.FTE.LOG.CMD.logger_name`. The second queue is a reject queue. Because the stand-alone database logger never discards log messages, if the logger encounters a message that it cannot handle, it places the message on the reject queue for examination, and possible reprocessing. You are not recommended to use the queue manager's dead letter queue for this purpose, because rejected messages do not have a DLH header and because rejected messages should not be combined with messages put to the dead letter queue for other reasons. The default name for the reject queue is `SYSTEM.FTE.LOG.RJCT.logger_name`. These two queues are defined in the MQSC script files generated by the **fteCreateLogger** command.
6. Choose a user and configure permissions
7. Optional: You can configure the stand-alone database logger further by editing the `logger.properties` file created by the **fteCreateLogger** command in step “2” on page 886. This file is a Java properties file that consists of key-value pairs. The `logger.properties` file is in the `MQ_DATA_PATH/mqft/config/coordination_qmgr_name/loggers/logger_name` directory. For more information about available properties and their effects, see [MFT logger configuration properties](#).

8. **Windows**

Optional: If you are using a Windows system, you can run the stand-alone database logger as a Windows service. Run the **fteModifyLogger** command with the **-s** parameter. For more information, see [fteModifyLogger](#).

- Optional: If the database being used is Oracle or you are connecting to a Db2 database remotely, you will need to specify a user name and password that the logger will use to authenticate with your database server. This user name and password is specified in a credentials file that conforms to the format defined by the `MQMFTCredentials.xsd` schema. For more information, see [MFT credentials file format](#). After creating the credential file, you must specify the location of the credentials file in the `logger.properties` file using the `wmqfte.database.credentials.file` property.
- Start the stand-alone database logger using the **fteStartLogger** command. By default, the stand-alone database logger runs in the background and the stand-alone database logger places output into a file in the `logs` directory. If you want to run the stand-alone database logger in the foreground and produce output to the console as well as to the log file, add the **-F** parameter to the **fteStartLogger** command.

If you carried out the previous step and used the **fteModifyLogger** command with the **-s** parameter on Windows, the stand-alone database logger starts as a Windows service.

Related tasks

[“Configuring user access for an MFT stand-alone database logger” on page 888](#)

In a test environment, you can add any new privileges needed to your normal user account. In a production environment, you are recommended to create a new user with the minimum permissions required to do the job.

Related reference

[MFT logger configuration properties](#)

[fteStartLogger](#)

[fteModifyLogger](#)

[Authorities for the MFT logger](#)

Using MFT with a remote database

You can use the Managed File Transfer logger to communicate with a database on a remote system.

About this task

If you have a database installed on a different machine from the machine Managed File Transfer is installed on, complete the following steps. The steps apply to both Db2 and Oracle unless stated otherwise.

Procedure

- Install a database client on the system that you have installed Managed File Transfer on.
- Add your remote database server to your local database client configuration. This configuration update is needed for Managed File Transfer and IBM MQ to correctly access the database.
- Specify the new properties in the `logger.properties` file to connect to the database using the credentials file: **wmqfte.database.credentials.file**.

Note:  Earlier versions of Managed File Transfer used the properties **wmqfte.oracle.user** or **wmqfte.database.user**, and **wmqfte.oracle.password** or **wmqfte.database.password**. These properties are now deprecated. Use **wmqfte.database.credentials.file** instead.

- Oracle only:** To allow a remote connection to the database, change the XAResourceManager stanza in the coordination queue manager's `qm.ini` file to the following (ensuring you change the database name, user name and user password to match your own information):

```
Oracle_XA+Acc=P/fte/og/  
qgw783jhT+SesTm=35+DB=FTEAUDIT1+SqlNet=FTEAUDIT1+threads=false,  
the change is highlighted in bold.
```

5. **Oracle only:** Specify a host and port in the `logger.properties` file, using the `wmqfte.oracle.host` and `wmqfte.oracle.port` properties. The default values for the host and port allow you to work with a local database client so if you have previously worked with a local database, you might not have set these values.

Related reference

[MFT logger configuration properties](#)

Configuring user access for an MFT stand-alone database logger

In a test environment, you can add any new privileges needed to your normal user account. In a production environment, you are recommended to create a new user with the minimum permissions required to do the job.

About this task

The number and type of user accounts you need to run the stand-alone database logger depends on the number of systems you use. You can install the stand-alone database logger, IBM MQ and your database on a single system, or across two systems. The stand-alone database logger must be on the same system as IBM MQ. The components can be installed in the following topologies:

Stand-alone database Logger, IBM MQ and the database all on the same system

You can define a single operating system user for use with all three components. This is a suitable configuration for the stand-alone database logger. The stand-alone database logger uses Bindings mode to connect to IBM MQ and a native connection to connect to the database.

Stand-alone database Logger and IBM MQ on one system, the database on a separate system

You create two users for this configuration: an operating system user on the system running the stand-alone database logger, and a operating system user with remote access to the database on the database server. This is a suitable configuration for the stand-alone database logger using a remote database. The stand-alone database logger uses Bindings mode to connect to IBM MQ and a client connection to access the database.

As an example, the rest of these instructions assume that the user is called `fte1og`, but you can use any user name. Configure the user's permissions as follows:

Procedure

1. Ensure that the user has permission to read and, where necessary, execute, the files installed as part of the Managed File Transfer Remote Tools and Documentation installation.
2. Ensure that the user has permission to create and write to any file in the `logs` directory (in the configuration directory). This directory is used for an event log, and if necessary for diagnostic trace and FFDC files.
3. Ensure that the user has its own group, and is not also in any groups with wide-ranging permissions on the coordination queue manager. The user should not be in the `mqm` group. On certain platforms, the `staff` group is automatically given queue manager access as well; the stand-alone database logger user should not be in the `staff` group. You can view authority records for the queue manager itself and for objects in it using the IBM MQ Explorer. Right-click the object and select **Object Authorities > Manage Authority Records**. At the command line, you can use the commands `dspmqaaut` (display authority) or `dmpmqaut` (dump authority).
4. Use the **Manage Authority Records** window in the IBM MQ Explorer or the `setmqaut` (grant or revoke authority) command to add authorities for the user's own group (on AIX, IBM MQ authorities are associated with groups only, not individual users). The authorities required are as follows:
 - Connect and Inquire on the queue manager (the IBM MQ Java libraries require Inquire permission to operate).
 - Subscribe permission on the `SYSTEM.FTE` topic.
 - Put permission on the `SYSTEM.FTE.LOG.RJCT.logger_name` queue.
 - Get permission on the `SYSTEM.FTE.LOG.CMD.logger_name` queue.

The reject and command queue names given are the default names. If you chose different queue names when you configured the stand-alone database logger queues, add the permissions to those queue names instead.

5. Perform the user configuration that is specific to the database you are using.

- If your database is Db2, carry out the following steps:

There are several mechanisms for managing database users with Db2. These instructions apply to the default scheme based on operating system users.

- Ensure that the `fte1log` user is not in any Db2 administration groups (for example, `db2iadm1`, `db2fadm1`, or `dasadm1`)
- Give the user permission to connect to the database and permission to select, insert and update on the tables that you created as part of [Step 2: create the required database tables](#)

- If your database is Oracle, carry out the following steps:

- Ensure that the `fte1log` user is not in any Oracle administration groups (for example, `ora_dba` on Windows or `dba` on AIX and Linux)
- Give the user permission to connect to the database and permission to select, insert and update on the tables that you created as part of [Step 2: create the required database tables](#)

Configuring MFT Db2 database logger for TLS

In this scenario, you can configure an IBM MQ Managed File Transfer (MFT) Db2 database logger to connect to a Db2 database by using TLS. This process establishes a secure communication between the MFT database logger and the Db2 database.

Before you begin

This task assumes that you have the following:

- A functioning MFT logger queue manager.
- An existing MFT database logger.
- Administrative access to the Db2 database.
- A properly configured Db2 instance with TLS enabled.

As an example, the rest of these instructions assume that the MFT logger queue manager is called `QM1`, and the database logger is called `LOGGER0`. For more information, see [TLS security protocols in IBM MQ](#) and [Using self-signed certificates for mutual authentication of a client and queue manager](#).

Configure the database logger for TLS as follows:

Procedure

1. Prepare the truststore TLS certificate and the keystore TLS certificate in the Public Key Cryptography Standards #12 (PKCS12) format. Save these certificate files as `trust.p12` and `key.p12` in the following directory: `/var/mqm/mqft/config/QM1/loggers/LOGGER0/`.
2. Create the `jcc.properties` file in the directory as `/var/mqm/mqft/config/QM1/loggers/LOGGER0/`. Add the following properties to the Db2 JCC properties file:

```
# SSL/TLS Section
db2.jcc.sslConnection=true
db2.jcc.sslTrustStoreLocation=/var/mqm/mqft/config/QM1/loggers/LOGGER0/trust.p12
db2.jcc.sslTrustStorePassword=YourTrustStorePassword
db2.jcc.sslKeyStoreLocation=/var/mqm/mqft/config/QM1/loggers/LOGGER0/key.p12
db2.jcc.sslKeyStorePassword=YourKeyStorePassword
db2.jcc.sslKeyStoreType=pkcs12
```

3. Set correct permissions to secure the `jcc.properties` file, the `trust.p12` certificate file, and the `key.p12` certificate file by using the following commands:

```
chmod 600 /var/mqm/mqft/config/QM1/loggers/LOGGER0/trust.p12
chown mqm:mqm /var/mqm/mqft/config/QM1/loggers/LOGGER0/trust.p12
```

```
chmod 600 /var/mqm/mqft/config/QM1/loggers/LOGGER0/key.p12
chown mqm:mqm /var/mqm/mqft/config/QM1/loggers/LOGGER0/key.p12
```

```
chmod 600 /var/mqm/mqft/config/QM1/loggers/LOGGER0/jcc.properties
chown mqm:mqm /var/mqm/mqft/config/QM1/loggers/LOGGER0/jcc.properties
```

4. Configure the JVM properties by setting the **MFT BFG_JVM_PROPERTIES** environment variable to include the Db2 JCC properties file:

```
BFG_JVM_PROPERTIES= "-Ddb2.jcc.propertiesFile=/var/mqm/mqft/config/QM1/loggers/LOGGER0/
jcc.properties"
export BFG_JVM_PROPERTIES
```

5. Restart the MFT logger to apply the configuration.
6. Verify the configuration by checking the MFT logger logs for any SSL/TLS connection errors and confirm that the connection to the Db2 database is established by using TLS.

Note: You can further customize the SSL/TLS configuration by adding more properties to the `jcc.properties` file as needed. For production environments, you need to make sure that your TLS certificates are securely managed.

Alternative configurations for an MFT stand-alone logger

Typically a Managed File Transfer stand-alone logger, whether it is a file or a database type, is on the same system as the coordination queue manager and is connected to the coordination queue manager in IBM MQ bindings mode. However, it can also be installed on the same system as any queue manager which has connectivity to the coordination queue manager. The stand-alone logger receives messages using a subscription, which the stand-alone logger creates automatically. This is the configuration described in the installation instructions.

However, if you have site-specific considerations, you can configure a stand-alone logger to receive messages in two other ways, controlled by the `wmqfte.message.source.type` property. This property is described in [MFT logger configuration properties](#).

Administrative subscription

By default, a stand-alone logger creates its own subscription to the `SYSTEM.FTE/Log/#` topic, using the default durable subscription options and a managed subscription (that is, the queue manager controls the backing queue used to hold the messages before they are passed to the application). If other options are required on the subscription or the queue, you can instead create a subscription yourself, set the options that you require, and configure the stand-alone logger to use that subscription instead. Remember to add permission for the stand-alone logger to use the subscription that you create.

An example of using this configuration is to partition the log space by using two wildcard subscriptions, to send logs from agents whose name begins with `FINANCE` into one database and logs from agents beginning with `ACCOUNTING` into another. This type of configuration requires two stand-alone logger instances, each with its own `logger.properties` file referring to the required subscription and its own command queue and reject queue.

To collect log messages only from agents whose names begin with `ACCOUNTING`, create a subscription object on your coordination queue manager with a topic string of `SYSTEM.FTE/Log/ACCOUNTING*`. Set the **Wildcard usage** value to **Character level wildcard**. You must also add entries to the

logger.properties file for your logger. For example, if you create a subscription object called ACCOUNTING.LOGS with these settings, add the following entries to the logger.properties file:

```
wmqfte.message.source.type=administrative_subscription
wmqfte.message.source.name=ACCOUNTING.LOGS
```

The stand-alone logger handles log messages that start with the topic string of SYSTEM.FTE/Log/ only. You can specify a more restrictive topic string, but you cannot specify a less restrictive string. If you do specify a less restrictive string in error, all publications that relate to a topic string other than SYSTEM.FTE/Log/ go to the reject queue, and the stand-alone logger produces the error message BFGDB0002E. This error message implies that there is a problem with the stand-alone logger configuration.

Queue

The typical topology is where the stand-alone logger runs on the same system as the coordination queue manager. If this is not possible, you can create a subscription on the coordination queue manager using a queue on another queue manager as the subscription destination (either using a remote queue definition or by using the DESTQMGR property of the subscription). The logger can then run on the system hosting the second queue manager and read the messages from the queue. To ensure transactional integrity, the stand-alone logger must always connect to its queue manager in bindings mode. You must define the reject queue and command queue on the same queue manager that the stand-alone logger connects to. The queue managers must be at IBM WebSphere MQ 7.5 or later.

For example, to collect log messages which are being placed on the queue USER.QUEUE by a subscription, add these entries to the logger.properties file:

```
wmqfte.message.source.type=queue
wmqfte.message.source.name=USER.QUEUE
```

Installing the Java EE database logger for MFT

Follow these instructions to install and configure the JEE database logger for use with Managed File Transfer.

About this task

For more information about the Java EE database logger, see the topic [“Configuring an MFT logger” on page 868](#).

Note: You cannot run a Java EE database logger at the same time as a stand-alone logger, unless these loggers are using separate instances of the database.

Procedure

1. Before installing the Java EE database logger, you must prepare your environment. Use the instructions in the topic [“Preparing to install the Java EE database logger for MFT” on page 892](#).
2. Install the Java EE database logger in a Java Platform, Enterprise Edition (Java EE) or Jakarta EE compliant application server.
For instructions, see [“Installing the Java EE database logger for MFT with WebSphere Application Server 9.0” on page 895](#)

Related tasks

[“Preparing to install the Java EE database logger for MFT” on page 892](#)

Follow these instructions to prepare your Managed File Transfer environment before installing the Java EE database logger.

[“Installing the Java EE database logger for MFT with WebSphere Application Server 9.0” on page 895](#)

Follow these instructions to install and configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer with WebSphere Application Server 9.0.

[“Configuring user access for the Java EE database logger for MFT” on page 899](#)

When you configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer, you need user accounts to access IBM MQ, your database, and your operating system. The number of operating system users that is required depend on the number of systems you are using to host these components.

[“Migrating from the stand-alone database logger to the Java EE database logger for MFT” on page 900](#)

You can migrate from the stand-alone database logger to the Java EE database logger. You must stop the stand-alone database logger and install the JEE database logger. To avoid losing or duplicating log entries you must stop messages being published to the SYSTEM.FTE topic before stopping the stand-alone database logger, and restart it after you have installed the Java EE database logger. Back up your database before migration.

Related reference

[Authorities for the MFT logger](#)

Preparing to install the Java EE database logger for MFT

Follow these instructions to prepare your Managed File Transfer environment before installing the Java EE database logger.

About this task

For more information about the Java EE database logger, see the topic [“Configuring an MFT logger” on page 868](#).

Procedure

1. Install your database software using the documentation for your database.
If JDBC support is an optional component for your database, you must install this component.
2. Create a database using the tools provided by your database. The database must have a table space and bufferpool page size of at least 8K.

The default schema name is FTELOG. If you use a schema name other than FTELOG, you must edit the provided SQL file appropriate to your database, `ftelog_tables_db2.sql` or `ftelog_tables_oracle.sql`, to reflect this before proceeding to the next step.

Note: The `ftelog_tables_db2.sql` and `ftelog_tables_oracle.sql` files are in the file path `<MQ-installation-path>/mqft/sql`

3. Create the required database tables using your database's tools.

 On [Multiplatforms](#), the files `ftelog_tables_db2.sql` and `ftelog_tables_oracle.sql` contain SQL commands that you can run to create the tables.

 On z/OS, the file that you need to run depends on the version of Db2 for z/OS that you are using:

- For Db2 for z/OS 9.0 and earlier, run the file `ftelog_tables_zos.sql` to create the tables. This file creates the tables using an INTEGER data type for fields which denote the sizes of files that are transferred and the table ID associated with each transfer.
 - For Db2 for z/OS 9.1 and later, run the file `ftelog_tables_zos_bigint.sql` to create the tables. This file creates the tables using a BIGINT data type for fields which denote the sizes of files that are transferred and the table ID associated with each transfer.
4. If you are using an Oracle database, add `USE_SID_AS_SERVICE_listener_name=on` to your `listener.ora` file, where `listener_name` is the name of the listener on your database server. Then restart the listener.

This parameter specifies that the database connection treats the Oracle system identifier (SID) as the service name for the database. This parameter is required because the MFT database logger connects

to Oracle databases only by using a SID but later Oracle databases (from version 12c) use a service name. For more information, see the documentation for your version of Oracle, for example [Oracle Net Listener Parameters in the listener.ora File](#). If you do not modify the `listener.ora` file, you might see the following message when you run the logger:

```
ORA-12505, TNS:listener does not currently know of SID given in connect descriptor
```

5. If you have changed the schema name from FTELOG, you must change the schema name in the EAR file. For more information, see [“Changing the schema name in the Java EE database logger for MFT” on page 893](#).
6. Create a reject queue in IBM MQ.
Because the logger never discards log messages, if the logger encounters a message that it cannot handle, it places the message on the reject queue for examination and possible reprocessing. Do not use the queue manager's dead letter queue for this purpose, because rejected messages do not have a DLH header and because rejected messages must not be combined with messages put to the dead letter queue for other reasons. The **fteCreateLogger** command creates a reject queue. The default name for this reject queue is `SYSTEM.FTE.LOG.RJCT.logger_name`.
7. Follow the instructions in the topic [“Configuring user access for the Java EE database logger for MFT” on page 899](#).

What to do next

Install the Java EE database logger in a Java EE or Jakarta EE compliant application server. Use the instructions in [“Installing the Java EE database logger for MFT with WebSphere Application Server 9.0” on page 895](#)

Changing the schema name in the Java EE database logger for MFT

The Java Platform, Enterprise Edition (Java EE) database logger can use a database that has a non-default schema name. You must change the schema name in the Java EE database logger EAR file.

About this task

To change the name of the schema that the Java EE database logger uses, complete the following steps:

Procedure

1. Extract the JPA JAR file from the EAR file by using the following command:

```
jar -xvf ear_file lib/jpa_file
```

where:

- *ear_file* is `com.ibm.wmqfte.databaseslogger.jee.oracle.ear` or `com.ibm.wmqfte.databaseslogger.jee.ear` depending on whether you are using Db2 or Oracle.
- *jpa_file* is `com.ibm.wmqfte.web.jpa.oracle.jar` or `com.ibm.wmqfte.web.jpa.jar` depending on whether you are using Db2 or Oracle.

2. Extract the `persistence.xml` file from the JPA JAR file by using the following command:

```
jar -xvf lib/jpa_file META-INF/persistence.xml
```

where:

- *jpa_file* is `com.ibm.wmqfte.web.jpa.oracle.jar` or `com.ibm.wmqfte.web.jpa.jar` depending on whether you are using Db2 or Oracle.

3. Edit the `persistence.xml` file to change the following line:

```
<property name="openjpa.jdbc.Schema" value="schema_name" />
```

where

- *schema_name* is the schema name you want to use.

4. Update JPA JAR with the modified *persistence.xml* file by using the following command:

```
jar -uvf lib/jpa_file META-INF/persistence.xml
```

where:

- *jpa_file* is *com.ibm.wmqfte.web.jpa.oracle.jar* or *com.ibm.wmqfte.web.jpa.jar* depending on whether you are using Db2 or Oracle.

5. Update the EAR file with the modified JPA JAR file by using the following command:

```
jar -uvf ear_file lib/jpa_file
```

where:

- *ear_file* is *com.ibm.wmqfte.databaselogger.jee.oracle.ear* or *com.ibm.wmqfte.databaselogger.jee.ear* depending on whether you are using Db2 or Oracle.
- *jpa_file* is *com.ibm.wmqfte.web.jpa.oracle.jar* or *com.ibm.wmqfte.web.jpa.jar* depending on whether you are using Db2 or Oracle.

What to do next

Use the modified EAR file to install the Java EE database logger.

Related tasks

[“Installing the Java EE database logger for MFT with WebSphere Application Server 9.0” on page 895](#)

Follow these instructions to install and configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer with WebSphere Application Server 9.0.

Setting the native library path in WebSphere Application Server 9.0

If you deploy the Java Platform, Enterprise Edition (Java EE) database logger application on WebSphere Application Server 9.0, and you want to use bindings mode connections between the application and IBM MQ, you must configure the IBM MQ messaging provider with the location of the IBM MQ native libraries on the system.

About this task

If you do not set the native library path in your application server, you might receive the following error message in the WebSphere Application Server 9.0 system out log:

```
A connection could not be made to WebSphere MQ for the following reason:
CC=2;RC=2495;AMQ8568: The native JNI library 'mqjbnd' was not found. [3=mqjbnd]
```

Use the WebSphere Application Server 9.0 administrative console to complete the following steps:

Procedure

1. In the navigation pane, expand **Resources > JMS > JMS Providers**.
2. Select the IBM MQ messaging provider that is at the correct scope for the connection factory or activation specification that creates the bindings mode connection.

Note: Native path information at **Server** scope is used in preference to native path information at higher scopes, and native path information at **Node** scope is used in preference to native path information at **Cell** scope.

3. Under **General Properties**, in the **Native library path** field, enter the full name of the directory that contains the IBM MQ native libraries.

For example, on Linux enter */opt/mqm/java/lib64*. Enter only one directory name.

4. Click **OK**.

Once the path is set, you should save the changes to the master configuration for the changes to take effect.

5. Restart the application server to refresh the configuration.
6. Required: Restart the application server a second time to load the libraries.

Related tasks

[“Installing the Java EE database logger for MFT with WebSphere Application Server 9.0” on page 895](#)
Follow these instructions to install and configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer with WebSphere Application Server 9.0.

Installing the Java EE database logger for MFT with WebSphere Application Server 9.0

Follow these instructions to install and configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer with WebSphere Application Server 9.0.

Before you begin

Before you install the JEE database logger application, follow the instructions in the topics [“Preparing to install the Java EE database logger for MFT” on page 892](#) and [“Setting the native library path in WebSphere Application Server 9.0” on page 894](#).

About this task

For more information about the Java EE database logger, see [“Configuring an MFT logger” on page 868](#).

Procedure

1. Set up the XA JDBC provider:
 - a) Select **Resources > JDBC > JDBC Providers** from the WebSphere Application Server 9.0 administration console navigation.
 - b) Create a JDBC provider using the console wizard, by clicking **New**.
 - c) At Step 1 of the wizard, select the database that you are using from the **Database type** list, and the associated provider type from the **Provider type** list. From the **Implementation type** list, select **XA data source**. Click **Next**.
 You can remove a reference to `db2jcc_license_cisuz.jar` and you should change `db2jcc.jar` to `db2jcc4.jar`, that is, the version of the jar file shipped with the latest version of Db2, or your local version.
 - d) At Step 2 of the wizard, ensure that the directory location of the required database jar files is set correctly. Click **Next**.
 - e) Click **Finish** on the summary page to create the JDBC provider.
2. Create authentication aliases. You create one alias for the data source and another for IBM MQ:
 - a) Select **Security > Global security** from the WebSphere Application Server 9.0 administration console navigation.
 - b) Under the **Authentication** heading, expand **Java Authentication and Authorization Service**.
 - c) Click **J2C authentication data**. The authentication alias page opens.
 - d) Create an authentication alias for your data source:
 - i) Click **New**.
 - ii) Enter the details for **Alias**, **User ID**, **Password**, and **Description**. The details that are entered in the **User ID** and **Password** fields must match the details that you entered when you created your database user. For more information, see [“Configuring user access for the Java EE database logger for MFT” on page 899](#).
 - iii) Click **OK**.
 - e) Create an authentication alias for IBM MQ:
 - i) Click **New**.

- ii) Enter the details for **Alias**, **User ID**, **Password**, and **Description**. The details that are entered in the **User ID** and **Password** fields must match your user and password settings for your IBM MQ installation.
 - iii) Click **OK**.
- 3. Create a data source:
 - a) Select **Resources > JDBC > Data sources** from the WebSphere Application Server 9.0 administration console navigation.
 - b) Select the **Scope** drop-down list and change the scope to the appropriate value. For example, Node=yourNode, Server=yourServer.
 - c) Create a data source using the console wizard, by clicking **New**.
 - d) At Step 1 of the wizard, in the **Data source name** field, enter wmqfte-database and in the **JNDI name** field, enter jdbc/wmqfte-database. Click **Next**.
 - e) At Step 2 of the wizard, use the **Select an existing JDBC provider** dropdown list to select the JDBC provider created in the previous steps. Click **Next**.
 - f) **Db2:** At Step 3 of the wizard, in the **Driver type** field, enter 4.
 - g) **Db2:** Enter the details in the **Database name**, **Server name**, and **Port number** fields, and click **Next**.

Oracle: Enter the connection URL in the **URL** field and choose the correct data store helper in the **Data store helper class name** field.

Oracle RAC: When connecting to an Oracle Real Application Cluster, the connection URL must include the host information necessary to connect to all available instances of the database.
 - h) At Step 4 of the wizard, select the name of the data source authentication alias that you defined in step 2d from the **Authentication alias for XA recovery** list. Select the same name from the **Component-managed authentication alias** and **Container-managed authentication alias** lists.
 - i) Click **Finish** on the summary page to create the data source.
- 4. Optional: Verify the configuration of the data source:
 - a) Select **Resources > JDBC > Data sources** from the WebSphere Application Server 9.0 administration console navigation.
 - b) Click the **Test Connection** button.
- 5. Create a topic.
 - a) From the WebSphere Application Server 9.0 administration console navigation, click **Resources > JMS > Topics**.
 - b) Select the **Scope** drop-down list and change the scope to the appropriate value. For example, Node=yourNode, Server=yourServer.
 - c) Click **New**.
 - d) Click **IBM MQ messaging provider**.
 - e) On the **Administration** panel of the property page for the topic, choose unique values for the **Name** and **JNDI name** fields, that you will reference later on in the configuration.
 - f) In the **IBM MQ topic** panel, enter SYSTEM.FTE/Log/# in the **Topic name** field.
- 6. Create an activation specification:
 - a) From the WebSphere Application Server 9.0 administration console navigation, click **Resources > JMS > Activation specifications**.
 - b) Select the **Scope** drop-down list and change the scope to the appropriate value. For example, Node=yourNode, Server=yourServer.
 - c) Click **New**.
 - d) Click **IBM MQ messaging provider**.
 - e) In Step 1 of the wizard, choose unique values for the **Name** and **JNDI name** fields, that you will again reference later on in the configuration.

- f) In Step 1.1, enter the JNDI name for the topic that you set up in step 5 in the **Destination JNDI name** field.
 - g) From the **Destination type** list, select **Topic**.
 - h) In Step 1.2 of the wizard, select **Durable Subscription**. Enter `SYSTEM.FTE.DATABASELOGGER.AUTO` in the **Subscription name** field.
 - i) In Step 2 of the wizard, select **Enter all the required information into this wizard**.
 - j) In Step 2.1, enter your queue manager name in the **Queue manager or queue sharing group name** field.
 - k) In Step 2.2, select your chosen transport method from the **Transport** list. If you select **Bindings**, no other information is required. If you select **Client** or **Bindings then client**, enter the details for **Hostname**, **Port**, and **Server connection channel**.
 - l) Optional: Click **Test Connection** to confirm the queue manager is present. However, you can expect to receive `NOT_AUTHORIZED` until you have referenced the authentication alias in step 6n.
 - m) Click **Save**.
 - n) Click the name of the Activation Specification that you created. In the **General Properties** section of the **Configuration** tab, scroll down to the **Advanced** panel and enter a unique name to identify your IBM MQ connection in the **Client ID** field. You must complete this step or your connection is rejected by IBM MQ with the `JMSCC0101` error code.
 - o) If you chose **Client** as your transport method, scroll down to the **Security Settings** panel and select the authentication alias that you defined in step 8 from the **Authentication alias** list.
 - p) Click **Apply**.
 - q) In the **Additional Properties** section of the **Configuration** tab, click **Advanced Properties**. In the **Connection Consumer** section of the **Advanced Properties** panel, enter 1 into the **Maximum server sessions** field.

Note: Ensure you complete this step before proceeding. Failure to do so can cause the logger to fail to operate correctly.
 - r) In the **Additional Properties** section of the **Configuration** tab, click **Advanced Properties**. Set the value of **Stop endpoint if message delivery fails** to a minimum of 1.

 If the value of the `_numberOfFailedAttemptsBeforeReject` property is set to more than 1 (see 9j for more information), set **Stop endpoint if message delivery fails** to at least the value of the `_numberOfFailedAttemptsBeforeReject` property. This prevents the endpoint from stopping when a message that cannot be processed (for example, a malformed transfer log message) is received. For more information, see [MFT logger error handling and rejection](#).
7. Create a queue connection factory.
- a) From the WebSphere Application Server 9.0 administration console navigation, click **Resources > JMS > Queue connection factories**.
 - b) Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.
 - c) Click **New**.
 - d) Click **IBM MQ messaging provider**.
 - e) In Step 1 of the wizard, choose unique values for the **Name** and **JNDI name** fields, that you will again reference later on in the configuration.
 - f) In Step 2, select **Enter all the required information into this wizard**.
 - g) In Step 2.1, enter your queue manager name in the **Queue manager or queue sharing group name** field.
 - h) In Step 2.2, select your chosen transport method from the **Transport** list. If you select **Bindings**, no other information is required. If you select **Client** or **Bindings then client**, enter the details for **Hostname**, **Port**, and **Server connection channel**.

- i) Optional: Click **Test Connection** to confirm the queue manager is present. However, you can expect to receive NOT_AUTHORIZED until you have referenced the authentication alias in step 7h.
 - j) If you selected **Client** or **Bindings then client** as your transport method, click the name of the queue connection factory you have just created. Scroll down to the **Security Settings** panel of the **Configuration** tab and select the authentication alias that you defined in step 2e from the **Authentication alias for XA recovery** and **Container-managed authentication alias** lists.
8. Create a reject queue in WebSphere Application Server:
- a) From the WebSphere Application Server 9.0 administration console navigation, click **Resources > JMS > Queues**.
 - b) Select the **Scope** drop-down list and change the scope to the appropriate value. For example, Node=yourNode, Server=yourServer.
 - c) Click **New**.
 - d) Click **IBM MQ messaging provider**.
 - e) Choose unique values for the **Name** and **JNDI name** fields, that you will again reference later on in the configuration.
 - f) Enter SYSTEM.FTE.LOG.RJCT.*logger_name* in the **Queue name** field. Ensure you have created this queue on your coordination queue manager.
 - g) Enter your queue manager name in the **Queue manager name** field.
 - h) Click **OK**.
9. Install the JEE database logger application:
- a) From the WebSphere Application Server 9.0 administration console, select **Applications > New Application**.
 - b) Select the **Scope** drop-down list and change the scope to the appropriate value. For example, Node=yourNode, Server=yourServer.
 - c) From the options list, select **New Enterprise Application**.
 - d) On the **Preparing for the application installation** page, select the com.ibm.wmqfte.databaselogger.jee.ear file or the com.ibm.wmqfte.databaselogger.jee.oracle.ear file from the MQ_INSTALLATION_PATH/mqft/web directory of the Managed File Transfer Service installation, and click **Next**.
 - e) On the following screen, select **Detailed** to show all installation options and parameters, and click **Next**.
 - f) Click **Next** through wizard steps 1-4 to accept the default values.
 - g) In step 5 of the wizard, **Bind listeners for message driven beans**, scroll to the **Listener Bindings** section. Click **Activation Specification**.
Enter the required values for the following fields:
Target Resource JNDI name
The JNDI name that you specified when creating an activation specification in step 6d.
Destination JNDI name
The JNDI name that you specified when creating a topic in step 5d.
Click **Next**.
 - h) In step 6 of the wizard, **Map resource references to resources**, enter the details in the **Target Resource JNDI name** field. This name is the JNDI name that you specified for the reject queue connection factory in step 7c. Click **Next**.
 - i) In step 7 of the wizard, **Map resource environment entry references to resources**, enter the details in the **Target Resource JNDI name** field. This name is the JNDI name of the reject queue that you created in step 8d. Click **Next**.
 - j) In step 8 of the wizard, **Map environment entries for EJB modules**, accept the default value of 1. Click **Next**.

Oracle RAC: When connecting to an Oracle Real Application Cluster you must set the value of the `_numberOfFailedAttemptsBeforeReject` property to **at least 2**. This property determines the number of times that the logger attempts to process an audit message after a failure occurs. In a case of database failover at least one failure is likely to occur. To avoid unnecessarily moving a message to the reject queue, increasing this value allows a second attempt to be made, which usually results in success as a connection is made to the new database instance. If you find during testing that messages are still moved to the reject queue during failover of your database instance, increase this value further: the timing of the switch between instances might cause more than one failure for the same message. However, be aware that increasing this value affects all failure cases (for example, a malformed message) and not just database failover, so increase the value with care to avoid unnecessary retries.

- k) In step 9 of the wizard, **Metadata for modules**, click **Next**.
 - l) In step 10 of the wizard, **Summary**, click **Finish**.
10. You can now start the application from the WebSphere Application Server 9.0 administration console:
- a) Select **Applications > Application Types > WebSphere enterprise applications** from the console navigation.
 - b) Select the check box for the **Logger** enterprise application from the collection table, and click **Start**.

Configuring user access for the Java EE database logger for MFT

When you configure the Java Platform, Enterprise Edition (Java EE) database logger for Managed File Transfer, you need user accounts to access IBM MQ, your database, and your operating system. The number of operating system users that is required depend on the number of systems you are using to host these components.

About this task

The number and type of user accounts you need to run the Java EE database logger depend on the number of systems you use. User accounts are required to access the following three environments:

- Local operating system
- IBM MQ
- Database

You can install the JEE database logger, IBM MQ and your database on a single system, or across several systems. The components can be installed in the following example topologies:

Java EE database logger, IBM MQ, and the database all on the same system

You can define a single operating system user for use with all three components. The logger uses Bindings mode to connect to IBM MQ and a native connection to connect to the database.

Java EE database logger and IBM MQ on one system, the database on a separate system

You create two users for this configuration: an operating system user on the system running the logger, and an operating system user with remote access to the database on the database server. The logger uses Bindings mode to connect to IBM MQ and a client connection to access the database.

Java EE database logger on one system, IBM MQ on another system, the database on a further system

You create three users for this configuration: An operating system user to start the application server, an IBM MQ user to access the queues and topics being used, and a database server user to access and insert into the database tables. The logger uses Client mode to access IBM MQ and a client connection to access the database.

As an example, the rest of these instructions assume that the user is called `fteLog`, but you can use any user name, new or existing. Configure the user permissions as follows:

Procedure

1. Ensure that the operating system user has its own group, and is not also in any groups with wide-ranging permissions on the coordination queue manager. The user should not be in the mqm group. On certain platforms, the staff group is automatically given queue manager access as well; the logger user should not be in the staff group. You can view authority records for the queue manager itself and for objects in it using the IBM MQ Explorer. Right-click the object and select **Object Authorities > Manage Authority Records**. At the command line, you can use the commands [dspmqaut](#) (display authority) or [dmpmqaut](#) (dump authority).
2. Use the **Manage Authority Records** window in the IBM MQ Explorer or the [setmqaut](#) (grant or revoke authority) command to add authorities for the IBM MQ user's own group (on AIX, IBM MQ authorities are associated with groups only, not individual users). The authorities required are as follows:
 - CONNECT and INQUIRE on the queue manager (the IBM MQ Java libraries require INQUIRE permission to operate).
 - SUBSCRIBE permission on the SYSTEM.FTE topic.
 - PUT permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.

The reject and command queue names given are the default names. If you chose different queue names when you configured the logger queues, add the permissions to those queue names instead.

3. Perform the database user configuration that is specific to the database you are using.
 - If your database is Db2, carry out the following steps:

Note: There are several mechanisms for managing database users with Db2. These instructions apply to the default scheme based on operating system users.

 - Ensure that the `fte1og` user is not in any Db2 administration groups (for example, `db2iadm1`, `db2fadm1`, or `dasadm1`).
 - Give the user permission to connect to the database and permission to select, insert, and update on the tables that you created as part of [Step 2: create the required database tables](#).
 - If your database is Oracle, carry out the following steps:
 - Ensure that the `fte1og` user is not in any Oracle administration groups (for example, `ora_dba` on Windows or `dba` on AIX and Linux).
 - Give the user permission to connect to the database and permission to select, insert and update on the tables that you created as part of [Step 2: create the required database tables](#).

Migrating from the stand-alone database logger to the Java EE database logger for MFT

You can migrate from the stand-alone database logger to the Java EE database logger. You must stop the stand-alone database logger and install the JEE database logger. To avoid losing or duplicating log entries you must stop messages being published to the SYSTEM.FTE topic before stopping the stand-alone database logger, and restart it after you have installed the Java EE database logger. Back up your database before migration.

About this task

Procedure

1. Before stopping the database, run the following MQSC command against your coordination queue manager: `ALTER QM PSMODE (COMPAT)`

This stops messages being published to the SYSTEM.FTE/Log topic. Wait until the logger has processed all of the messages on its subscription. By default, this subscription is called `SYSTEM.FTE.LOGGER.AUTO`.
2. Stop the database logger using the **fteStopLogger** command.

3. Back up the database using the tools supplied with the database software.
4. Delete the subscription belonging to the stand-alone database logger.

By default, this subscription is called SYSTEM.FTE.LOGGER.AUTO.

5. If your database schema is at an earlier version, you must migrate the schema to each subsequent level in order. For example, if your database schema is at V7.0.1 and you are migrating to V7.0.4, you must migrate your schema from V7.0.1 to V7.0.2, then from V7.0.2 to V7.0.3, and then from V7.0.3 to V7.0.4. Migrate your database schema from version *old* to version *new*, where *old* and *new* are variables that describe a schema version, by performing the one of the following actions for each version of the schema that you must migrate through:
 -  If your database is Db2 on z/OS and you are migrating between the V7.0.2 and V7.0.3 schemas or between the V7.0.3 and V7.0.4 schemas, you must create a new database schema and copy your existing data into it. For more information, see the Db2 documentation.
 - If your database is not Db2 or you created your database with a page size of more than 8K, you can migrate the schema in the same way as for other versions, by completing the following steps.
 - If you are migrating between database tables in any other circumstances complete the following steps:
 - a. Choose the file that is appropriate to your database platform and has a name that includes the string *old-new*. This file is located in the `MQ_INSTALLATION_PATH/mqft/sql` directory of the Remote Tools and Documentation installation.
 - b. If you have made modifications to the initial schema, review the migration file to ensure that the file will be compatible with your modified database.
 - c. Run the SQL file against your database.
6. Install the Java EE database logger EAR file.
7. Deploy the Java EE database logger. For more information, see [“Installing the Java EE database logger for MFT”](#) on page 891.
8. Run the following MQSC command against your coordination queue manager: `ALTER QMGR PSMODE(ENABLED)`
This enables publishing of messages to the SYSTEM.FTE/Log topic.

Results

Configuring the Connect:Direct bridge

Configure the Connect:Direct bridge to transfer files between a Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Before you begin

The agent and node that make up the Connect:Direct bridge must be on the same system, or have access to the same file system, for example through a shared NFS mount. This file system is used to temporarily store files during file transfers that involve the Connect:Direct bridge, in a directory defined by the `cdTmpDir` parameter. The Connect:Direct bridge agent and the Connect:Direct bridge node must be able to address this directory using the same path name. For example, if the agent and node are on separate Windows systems, the systems must use the same drive letter to mount the shared file system. The following configurations allow the agent and the node to use the same path name:

- The agent and node are on the same system, which is either running Windows or Linux for x86-64
- The agent is on Linux for x86-64, and the node is on AIX
- The agent is on one Windows system, and the node is on another Windows system

The following configurations do not allow the agent and the node to use the same path name:

- The agent is on Linux for x86-64, and the node is on Windows
- The agent is on Windows, and the node is on UNIX

Consider this restriction when planning your installation of the Connect:Direct bridge.

For more details of the operating system versions supported for the Connect:Direct bridge, see the web page [System Requirements for IBM MQ](#).

About this task

A Connect:Direct bridge agent is a Managed File Transfer agent that is dedicated to communicating with a Connect:Direct node.

By default, the Connect:Direct bridge agent uses the TCP/IP protocol to connect to the Connect:Direct node. If you want a secure connection between your Connect:Direct bridge agent and the Connect:Direct node, you can use the SSL protocol or the TLS protocol.

Procedure

1. Choose the operating systems for the Connect:Direct bridge agent and node:
 - a) Choose a system running either Windows or Linux on x86-64 to install the Connect:Direct bridge agent on.
 - b) Choose an operating system that is supported by Connect:Direct for Windows or Connect:Direct for UNIX to install the Connect:Direct bridge node on.
2. Choose and configure a Connect:Direct node.

You must have a Connect:Direct node installed before following these instructions.

- a) Choose a Connect:Direct node for the Managed File Transfer agent to communicate with.
- b) Check the network map for your chosen Connect:Direct node. If the network map contains any entries for remote nodes running on a Windows operating system, you must ensure that these entries specify that the nodes are running on Windows.



If the Connect:Direct node that you have selected for the Connect:Direct bridge is running on Windows, use the Connect:Direct Requester to edit the network map. Ensure that the **Operating System** field for any remote nodes that are running on Windows is set to **Windows**.

3. Create and configure a Connect:Direct bridge agent.
 - a) Create a Connect:Direct bridge agent using the **fteCreateCDAgent** command.
 - You must provide a value for the **cdNode** parameter. This parameter specifies the name that the agent uses for the Connect:Direct node that is part of the Connect:Direct bridge. Use the name of the Connect:Direct node that you chose in the previous section.
 - Provide values for the **cdNodeHost** and **cdNodePort** parameters, which define the Connect:Direct node that the agent communicates with.

If you do not provide a value for the **cdNodeHost** parameter, the host name or IP address of the local system is used. If you do not provide a value for the **cdNodePort** parameter, the value 1363 is used.
 - Optionally, use the information in [fteCreateAgent](#) to determine whether you need to specify a value for the **cdTmpDir** parameter.
 - b) Map the user credentials used by Managed File Transfer to user credentials on a Connect:Direct node. You can map credentials by using one of the following methods:
 - Create a `ConnectDirectCredentials.xml` file to define credential mapping information. For more information, see [“Mapping credentials for Connect:Direct by using the ConnectDirectCredentials.xml file”](#) on page 904.

- Write a user exit to perform credential mapping for your Connect:Direct bridge. For more information, see [“Mapping credentials for Connect:Direct by using exit classes”](#) on page 906.
4. Configure the `ConnectDirectNodeProperties.xml` file to include information about the remote Connect:Direct nodes.

You must have created a Connect:Direct bridge agent before following these instructions.

Edit the template `ConnectDirectNodeProperties.xml` in the Connect:Direct bridge agent configuration directory. For each Connect:Direct node or group of nodes that you want to define information about, perform the following steps:

- a) Inside the `nodeProperties` element, create a `node` element.
- b) Add a `name` attribute to the `node` element. Specify the value of this attribute as a pattern to match the name of one or more remote Connect:Direct nodes.
- c) Optional: Add a `pattern` attribute to the `node` element that specifies what type of pattern the value in the `name` attribute is. Valid values are `regex` and `wildcard`. The default option is `wildcard`.
- d) Add a `type` attribute to the `node` element that specifies the operating system that the remote Connect:Direct nodes specified by the `name` attribute run on.

The following values are valid:

- Windows - the node runs on Windows
- UNIX - the node runs on AIX and Linux
-  z/OS, zos, os/390, or os390 - the node runs on z/OS

The value of this attribute is not case sensitive. Transfers to remote nodes on other operating systems are not supported by the Connect:Direct bridge.

For more information, see [Connect:Direct node properties file format](#).

5. Configure a secure connection between the Connect:Direct bridge agent and the Connect:Direct node. For an example of how to do this, see [Configuring SSL or TLS between the Connect:Direct bridge agent and the Connect:Direct node](#).

Related tasks

[Troubleshooting the Connect:Direct bridge](#)

[Configuring SSL or TLS between the Connect:Direct bridge agent and the Connect:Direct node](#)

[Transferring a file to a Connect:Direct node](#)

[Transferring a file from a Connect:Direct node](#)

 [Transferring multiple files from a Connect:Direct node](#)

Related reference

[The Connect:Direct bridge](#)

Mapping credentials for Connect:Direct

Map user credentials in Managed File Transfer to user credentials on a Connect:Direct node by using the default credential mapping function of the Connect:Direct bridge agent or by writing your own user exit. Managed File Transfer provides a sample user exit that performs user credential mapping.

Related tasks

[“Mapping credentials for Connect:Direct by using the ConnectDirectCredentials.xml file”](#) on page 904

Map user credentials in Managed File Transfer to user credentials on Connect:Direct nodes by using the default credential mapping function of the Connect:Direct bridge agent. Managed File Transfer provides an XML file that you can edit to include your credential information.

[“Mapping credentials for Connect:Direct by using exit classes”](#) on page 906

If you do not want to use the default credential mapping function of the Connect:Direct bridge agent, you can map user credentials in Managed File Transfer to user credentials on a Connect:Direct node by writing

your own user exit. Configuring your own credential mapping user exits disables the default credential mapping function.

Related reference

[CDCredentialExit.java interface](#)

[Connect:Direct credentials file format](#)

Mapping credentials for Connect:Direct by using the *ConnectDirectCredentials.xml* file

Map user credentials in Managed File Transfer to user credentials on Connect:Direct nodes by using the default credential mapping function of the Connect:Direct bridge agent. Managed File Transfer provides an XML file that you can edit to include your credential information.

About this task

After a Connect:Direct bridge agent has been created using the **fteCreateCDAgent** command, a *ConnectDirectCredentials.xml* file needs to be manually created. Before you can use a Connect:Direct bridge agent, you must edit this file to include host, user, and credential information. For more information, see [Connect:Direct credentials file format](#). By default, this file is loaded from the current user's home directory, */home/fteuser/ConnectDirectCredentials.xml* for example. To use another location, specify it using the `<credentialsFile>` element in the *ConnectDirectNodeProperties.xml* file.

Procedure

1. Ensure that the name attribute in the element `<tns:pnode name="Connect:Direct node host" pattern="wildcard">` contains the value of the name of the Connect:Direct node that the Connect:Direct bridge agent connects to. This value must be the same value that you specify for the **fteCreateCDAgent -cdNode** parameter.
The value of the pattern attribute can be either wildcard or regex. If this attribute is not specified, the default is wildcard.
2. Insert user ID and credential information into the file as child elements of `<tns:pnode>`.
You can insert one or more instances of the following `<tns:user>` element into the file:

```
<tns:user name="name"
          pattern="pattern"
          ignorecase="ignorecase"
          cdUserId="cdUserId"
          cdPassword="cdPassword"
          pnodeUserId="pnodeUserId"
          pnodePassword="pnodePassword">
</tns:user>
```

where:

- *name* is a pattern to match the MQMD user ID associated with the MFT transfer request.
- *pattern* specifies whether the pattern specified for the name attribute is a wildcard expression or a Java regular expression. The value of the pattern attribute can be either wildcard or regex. If this attribute is not specified, the default is wildcard.
- *ignorecase* specifies whether to treat the pattern specified by the name attribute as case sensitive. If this attribute is not specified, the default is true.
- *cdUserId* is the user ID that is used by the Connect:Direct bridge agent to connect to the Connect:Direct node specified by the name attribute of `<tns:pnode>` element. If possible, ensure that *cdUserId* is a Connect:Direct administrator user ID. If *cdUserId* cannot be a Connect:Direct administrator, ensure that the user ID has the following functional authorities at the Connect:Direct bridge node:

- For a Windows node set the following authorities. This example is formatted with carriage returns to aid readability:

```
View Processes in the TCQ      value: yes
Issue the copy receive, copy send, run job, and run task Process statements
Issue the submit Process statement      value: yes
Monitor, submit, change, and delete all Processes      value: all
Access Process statistics      value: all
Use the trace tool or issue traceon and traceoff commands      value: yes
Override Process options such as file attributes and remote node ID      value: yes
```

- For an AIX or Linux node set the following parameters in the `userfile.cfg` file:

```
pstmt.copy      value: y
pstmt.upload    value: y
pstmt.download  value: y
pstmt.runjob    value: y
pstmt.runtask   value: y
cmd.submit      value: y
pstmt.submit    value: y
cmd.chgproc     value: y
cmd.delproc     value: y
cmd.flsproc     value: y
cmd.selproc     value: a
cmd.selstats    value: a
cmd.trace       value: y
snode.ovrd      value: y
```

- `cdPassword` is the password associated with the user ID specified by the `cdUserId` attribute.
- You can optionally specify the `pnodeUserId` attribute. The value of this attribute is the user ID that is used by the Connect:Direct node specified by the name attribute of `<tns:pnode>` element to submit the Connect:Direct process. If you do not specify the `pnodeUserId` attribute, the Connect:Direct node uses the user ID specified by the `cdUserId` attribute to submit the Connect:Direct process.
- You can optionally specify the attribute `pnodePassword`. The value of this attribute is the password associated with the user ID specified by the `pnodeUserId` attribute.

If no user element matches the MQMD user ID, the transfer fails.

3. Optional: You can include one or more `<tns:snode>` elements as child elements of the `<tns:user>` element. The `<tns:snode>` element specifies credentials that are used by the Connect:Direct node that is part of the Connect:Direct bridge. These credentials are the user ID and password that the Connect:Direct bridge node uses to connect to the Connect:Direct node that is the source or destination of the file transfer.

Insert one or many of the following elements into the file:

```
<tns:snode name="name"
           pattern="pattern"
           userId="userId"
           password="password" />
```

where:

- `name` is a pattern to match the name of the Connect:Direct node that is the source or destination of the file transfer.

- *pattern* specifies whether the pattern specified for the name attribute is a wildcard expression or a Java regular expression. The value of the pattern attribute can be either *wildcard* or *regex*. If this attribute is not specified, the default is *wildcard*.
- *userId* is the user ID that is used by the Connect:Direct node specified by the name attribute of the <tns:pnode> element to connect to a Connect:Direct node that matches the pattern specified by the name attribute of <tns:snode>.
- *password* is the password associated with the user ID specified by the *userId* attribute.

If no <tns:snode> element matches the secondary node of the file transfer, this does not cause the transfer to fail. The transfer is started and no user ID and password are specified for use with the snode.

Results

When searching for a pattern match for user names or Connect:Direct node names the Connect:Direct bridge agent searches from the start of the file to the end of the file. The first match that is found is the one that is used.

Related tasks

[“Configuring the Connect:Direct bridge” on page 901](#)

Configure the Connect:Direct bridge to transfer files between a Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference

[Connect:Direct credentials file format](#)

[fteCreateCDAgent: create a Connect:Direct bridge agent](#)

Mapping credentials for Connect:Direct by using exit classes

If you do not want to use the default credential mapping function of the Connect:Direct bridge agent, you can map user credentials in Managed File Transfer to user credentials on a Connect:Direct node by writing your own user exit. Configuring your own credential mapping user exits disables the default credential mapping function.

About this task

User exits that you create for mapping Connect:Direct credentials must implement the interface `com.ibm.wmqfte.exitroutine.api.ConnectDirectCredentialExit`. For more information, see [CDCredentialExit.java interface](#).

Configuring the IBM MQ Console and REST API

The mqweb server that hosts the IBM MQ Console and REST API is provided with a default configuration. In order to use either of these components a number of configuration tasks need to be completed, such as configuring security to allow users to log in. This topic describes all the configuration options that are available.

Procedure

- [“Basic configuration for the mqweb server” on page 907](#)
- [“Configuring security” on page 912](#)
- [“Configuring the HTTP host name” on page 912](#)
- [“Configuring the HTTP and HTTPS ports” on page 913](#)
- [“Configuring the response timeout” on page 915](#)
- [“Configuring autostart” on page 916](#)
- [“Configuring logging” on page 917](#)

- [“Configuring the LTPA token” on page 920](#)
- [“Configuring remote queue manager connection behavior for the IBM MQ Console” on page 922](#)
- [“Configuring the administrative REST API gateway” on page 924](#)
- [“Configuring the messaging REST API” on page 925](#)
- [“Configuring the REST API for MFT” on page 931](#)
- [“Tuning the mqweb server JVM” on page 937](#)
- [“File structure of the IBM MQ Console and REST API installation component” on page 938](#)

Basic configuration for the mqweb server

Before you can start to use the REST API or IBM MQ Console, you must install the correct components and configure the mqweb server that hosts the REST API or IBM MQ Console.

About this task

The procedure for this task focuses on a basic configuration for the mqweb server so that you can get started quickly with the REST API and IBM MQ Console. The steps for configuring security outline how to set up a basic user registry, but other options for configuring users and roles exist. For more information about configuring security for the mqweb server, see [IBM MQ Console and REST API security](#).

Note: You must have access to the `mqwebuser.xml` file to complete this procedure:

-  On z/OS, you must be a user that has write access to the `mqwebuser.xml` file.
-  On all other operating systems, you must be a [privileged user](#) to access the `mqwebuser.xml` file.
-  If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the `mqwebuser.xml` file in the IBM MQ Web Server data directory.

Procedure

1. Install the IBM MQ Console and REST API component:

-  On AIX, install the `mqm.web.rte` fileset. For more information about installing filesets on AIX, see [AIX installation tasks](#).
-  On IBM i, install the WEB component. To use this feature, you must also install the 5724L26 IBM MQ Java Messaging and Web Services, and 5770JV1 Java SE 8 prerequisites. For more information about installing features on IBM i, see [IBM i installation tasks](#).
-  On Linux, install the MQSeriesWeb component. For more information about installing components on Linux, see [Linux installation tasks](#).

From IBM MQ 9.4.0, you can also run the mqweb server in a stand-alone IBM MQ Web Server installation on Linux. For more information, about installing the IBM MQ Web Server, see [Installing the stand-alone IBM MQ Web Server](#).

-  On Windows, install the Web Administration feature. For more information about installing features on Windows, see [Windows installation tasks](#).
-  Install the IBM MQ for z/OS UNIX System Services Web Components feature. For more information about installing components and features on z/OS, see [z/OS installation tasks](#).

Note:  From IBM MQ 10.0, the Java runtime environment (JRE) on z/OS has support for IBM Semeru Runtime 21.

2. Create the mqweb server that hosts the IBM MQ Console and REST API.

-  On z/OS, run the **crtmqweb** script.
This script creates the WebSphere Liberty user directory that contains the mqweb server configuration and log files. For more information about running the **crtmqweb** script, see [“crtmqweb \(creating the mqweb server\)” on page 1018](#).
 -  In a stand-alone IBM MQ Web Server installation, follow the steps in [“Configuring the stand-alone IBM MQ Web Server” on page 911](#).
 - In all other environments, you do not need to complete any actions to create the mqweb server.
3.  On z/OS, create a cataloged procedure to start the mqweb server.
For more information, see [“Creating a procedure for the mqweb server” on page 1020](#).
4. Replace the existing configuration file, `mqwebuser.xml` with the basic registry sample file that is configured to offer basic security. Copy the `basic_registry.xml` file from the `MQ_INSTALLATION_PATH/web/mq/samp/configuration` directory to the appropriate directory for your system, and rename the file to `mqwebuser.xml`:
- In an IBM MQ installation, copy the file to the following directory:
 -   On AIX and Linux: `/var/mqm/web/installations/installationName/servers/mqweb`
 -  On Windows:
`MQ_DATA_PATH\web\installations\installationName\servers\mqweb`
Where `MQ_DATA_PATH` is the IBM MQ data path, this path is the data path that is selected during the installation of IBM MQ. By default, this path is `C:\ProgramData\IBM\MQ`.
 -  On z/OS: `WLP_user_directory/servers/mqweb`
Where `WLP_user_directory` is the directory that was specified when the **crtmqweb** script ran to create the mqweb server definition.
 -  In a stand-alone IBM MQ Web Server installation:
`MQ_OVERRIDE_DATA_PATH/web/installations/MQWEBINST/servers/mqweb`
where `MQ_OVERRIDE_DATA_PATH` is the IBM MQ Web Server data directory that the **MQ_OVERRIDE_DATA_PATH** environment variable points to.

The `basic_registry.xml` sample file configures four users:

mqadmin

An administrative user that is a member of the MQWebAdmin role.

mqreader

A read-only administrative user that is a member of the MQWebAdminRO role.

mftadmin

An administrative user that is a member of the MFTWebAdmin role.

mftreader

A read-only administrative user that is a member of the MFTWebAdminRO role.

All users are also members of the MQWebUser role.

For more information about the available roles, see [Roles on the IBM MQ Console and REST API](#)

5. Optional: Edit the `mqwebuser.xml` file to add more users and groups. Assign those users and groups appropriate roles to be authorized to use the REST API or IBM MQ Console. You can also change the passwords for the users that are defined by default, and encode the new passwords. For more information, see [Configuring users and roles](#).

Note:

- ▶ **z/OS** On z/OS, if you add users to the MQWebUser role, you must also grant the mqweb started task user ID alternate user access to the user IDs with the MQWebUser role. For example:

```
RDEFINE MQADMIN hlq.ALTERNATE.USER.userId UACC(NONE)
PERMIT hlq.ALTERNATE.USER.userId CLASS(MQADMIN) ACCESS(UPDATE) ID(mqwebUserId)
```

- ▶ **Multi** ▶ **z/OS** To complete the steps for getting started with the messaging REST API, you must add a user to the mqwebuser.xml file. This user must have the same name as an existing IBM MQ user on your system. Following the same format as the other users in the XML file, add the user ID and a password after the following line in the XML file: `<user name="mftreader" password="mftreader"/>`.

6. Set your environment to point to the mqweb server configuration.

- ▶ **z/OS** On z/OS, set the WLP_USER_DIR environment variable so that the variable points to your mqweb server configuration, by entering the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

Where *WLP_user_directory* is the name of the directory that is passed to the **crtmqweb** command. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [“crtmqweb \(creating the mqweb server\)”](#) on page 1018.

- ▶ **Linux** In a stand-alone IBM MQ Web Server installation, set the **MQ_OVERRIDE_DATA_PATH** environment variable to the IBM MQ Web Server data directory. For example, if you chose to use `/var/mqweb` as your IBM MQ Web Server data directory, issue the following command:

```
export MQ_OVERRIDE_DATA_PATH=/var/mqweb
```

- In all other environments, you do not need to complete any actions to set your environment.

7. By default, REST API and IBM MQ Console are available only from the same host as the mqweb server. Enable remote connections to the mqweb server by entering the following command:

```
setmqweb properties -k httpHost -v hostname
```

Where *hostname* specifies the IP address, domain name server (DNS) host name with domain name suffix, or the DNS host name of the server where IBM MQ is installed. Use an asterisk, *, in double quotation marks, to specify all available network interfaces, as shown in the following example:

```
setmqweb properties -k httpHost -v "*"
```

8. Optional: By default, the administrative REST API for MFT is not enabled. If you want to use this feature, you must enable it and configure a co-ordination queue manager:

- a) Enable the administrative REST API for MFT by entering the following command:

```
setmqweb properties -k mqRestMftEnabled -v true
```

- b) Configure which queue manager is the co-ordination queue manager by entering the following command:

```
setmqweb properties -k mqRestMftCoordinationQmgr -v qmgrName
```

Where *qmgrName* is the name of the coordination queue manager.

- c) To enable POST calls, configure which queue manager is the command queue manager by entering the following command :

```
setmqweb properties -k mqRestMftCommandQmgr -v qmgrName
```

Where *qmgrName* is the name of the command queue manager.

9. Start the mqweb server that supports the REST API and IBM MQ Console:

- ▶ **ALW** On AIX, Linux, and Windows, as a privileged user, enter the following command:

```
stmqweb
```

- ▶ **IBM i** On IBM i, as a privileged user, enter the following command in Qshell:

```
/QIBM/ProdData/mqm/bin/stmqweb
```

- ▶ **z/OS** On z/OS, start the procedure that you created in [“Creating a procedure for the mqweb server”](#) on page 1020.

The following messages are issued to the STDOUT DD to indicate that the mqweb server has started successfully.

```
[AUDIT ] MQWB2019I: MQ Console level: 9.2.4 - V924-CD924-L211028
[AUDIT ] MQWB0023I: MQ REST API level: 9.2.4 - V924-CD924-L211028
[AUDIT ] CWWKZ0001I: Application com.ibm.mq.rest started in 1.763 seconds.
[AUDIT ] CWWKZ0001I: Application com.ibm.mq.console started in 2.615 seconds.
[AUDIT ] CWWKF0011I: The mqweb server is ready to run a smarter planet. The mqweb
server started in 10.016 seconds.
```

You can stop the mqweb server at any time by stopping the mqweb server started task on z/OS, or by using the **endmqweb** command. However, if the mqweb server is not running, you cannot use the REST API or IBM MQ Console.

10. ▶ **z/OS**

Optional: On z/OS, if you want to allow system automation products to trap the MQWB2019I and MQWB0023I messages that are issued when the IBM MQ Console and REST API start, configure the mqweb server to write these messages to the MVS console. To configure the mqweb server to write the MQWB2019I and MQWB0023I messages to the MVS console, edit the `mqwebuser.xml` file that you created in step [“4”](#) on page 908, and add the following line to the file:

```
<zosLogging enableLogToMVS="true" wtoMessage="MQWB2019I,MQWB0023I"/>
```

For more information about configuring z/OS Logging in the mqweb server, see [z/OS Logging \(zosLogging\)](#).

Note: ▶ **UNIX** Ensure that the correct language pack is installed for your system. The environment variable `LC_ALL` might need to be set to correctly display web server command messages in the appropriate language. For example:

```
export LC_ALL=fr_FR.UTF-8
dspmqweb properties
```

What to do next

1. Configure mqweb server settings, including enabling HTTP connections, and changing the port number. For more information, see [“Configuring the IBM MQ Console and REST API”](#) on page 906.
2. Optionally, configure the REST API:
 - a. Configure Cross Origin Resource Sharing for the REST API. By default, you cannot access the REST API from web resources that are not hosted on the same domain as the REST API. That is, cross-origin requests are not enabled. You can configure Cross Origin Resource Sharing (CORS) to allow cross-origin requests from specified URLs. For more information, see [Configuring CORS for the REST API](#).

- b. Configure the REST API for MFT. For more information, see [“Configuring the REST API for MFT” on page 931](#).
3. Use the REST API or IBM MQ Console:
 - [Getting started with the administrative REST API](#)
 - [Getting started with the messaging REST API](#)
 - [Getting started with the IBM MQ Console](#)

Linux

Configuring the stand-alone IBM MQ Web Server

From IBM MQ 9.4.0, you can run the mqweb server that hosts the IBM MQ Console and REST API in a stand-alone IBM MQ Web Server installation.

Before you begin

The stand-alone IBM MQ Web Server is available only on Linux.

Before you can configure the mqweb server, you must install the IBM MQ Web Server by following the steps in [Installing the stand-alone IBM MQ Web Server](#).

About this task

Follow the procedure in this task to create and configure a new mqweb server that runs in a stand-alone IBM MQ Web Server installation. You can configure more than one mqweb server to run in a stand-alone IBM MQ Web Server installation by repeating this procedure.

Procedure

1. Create the IBM MQ Web Server data directory.

The data directory is used to store the configuration and log files for the mqweb server that runs the IBM MQ Console and REST API. You can use any directory that you choose as the IBM MQ Web Server data directory.

The user ID that you use to start the mqweb server must be granted read and write access to the data directory.
2. Set the **MQ_OVERRIDE_DATA_PATH** environment variable to the data directory that you created in step “1” on page 911.

For example, if you chose to use /var/mqweb as your IBM MQ Web Server data directory, issue the following command:

```
export MQ_OVERRIDE_DATA_PATH=/var/mqweb
```
3. Use the **setmqenv** command to set up the IBM MQ environment.

Change to the bin directory of the IBM MQ Web Server installation directory, then issue the following command:

```
./setmqenv -s
```
4. Use the **crtmqdir** command to create the IBM MQ directories and files, in the data directory. The files that are created include a template definition for the mqweb server.

Issue the following command:

```
crtmqdir -s -f
```
5. Optional: If this mqweb server is the first that you created to run with this installation of the stand-alone IBM MQ Web Server, use the **mqlicense** command to review and accept the IBM MQ license. You must run this command as a user that has write access to the IBM MQ Web Server installation directory.

For example, issue the following command to view the IBM MQ license:

```
mqlicense
```

For more information, see [mqlicense](#).

6. Optional: To migrate an existing mqweb server to run in the newly configured stand-alone IBM MQ Web Server installation, complete the following steps:

- a. Back up your existing mqweb server configuration.
- b. Restore the files into the `MQ_OVERRIDE_DATA_PATH/web/installations/MQWEBINST` directory, where `MQ_OVERRIDE_DATA_PATH` is the IBM MQ Web Server data directory that you created in step “1” on page 911.

For more information, see [“Backing up and restoring your mqweb server configuration” on page 940](#).

Note: Some features of the IBM MQ Console and REST API are not available in a stand-alone IBM MQ Web Server installation. If you migrate a mqweb server from an IBM MQ installation to a stand-alone IBM MQ Web Server installation, these features cannot be used after migration. For more information about the restrictions that apply in a stand-alone IBM MQ Web Server installation, see [The IBM MQ Console and REST API](#).

What to do next

Configure the mqweb server by following the steps that are described in [“Basic configuration for the mqweb server” on page 907](#).

Configuring security

You can configure security for the IBM MQ Console and the REST API by editing the `mqwebuser.xml` file. You can configure and authenticate users by configuring either a basic user registry, or an LDAP registry, or any of the other registry types that are provided with WebSphere Liberty. You can then authorize those users by assigning users and groups a role.

About this task

To configure security for the IBM MQ Console, and REST API, you must configure users and groups. These users and groups can then be authorized to use the IBM MQ Console, or REST API, or both. For more information about configuring users and groups, and authenticating and authorizing users, see [IBM MQ Console and REST API security](#).

When users authenticate with the IBM MQ Console, an LTPA token is generated. This token enables the user to use the IBM MQ Console without re-authenticating until the token expires.

If you use token based authentication with the REST API, a different LTPA token is generated when the user logs in using the `/login` REST API resource with the HTTP POST method. You can configure when this token expires, and whether this token can be used for both HTTP and HTTPS connections. For more information, see [“Configuring the LTPA token” on page 920](#).

Procedure

- [IBM MQ Console and REST API security](#)
- [“Configuring the LTPA token” on page 920](#)

Configuring the HTTP host name

By default, the mqweb server which hosts the IBM MQ Console and REST API is configured to allow only local connections. That is, the IBM MQ Console and REST API can be accessed only on the system on

which the IBM MQ Console and REST API are installed. You can configure host name to allow remote connections by using the **setmqweb** command.

Before you begin

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  **z/OS** On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  **Multi** On all other operating systems, you must be a [privileged user](#).
-  **Linux** If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the `mqwebuser.xml` file in the IBM MQ Web Server data directory.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ_OVERRIDE_DATA_PATH** environment variable to the IBM MQ Web Server data directory.

Procedure

- View the current configuration of the HTTP host name by using the following command:

```
dspmqweb properties -a
```

The `httpHost` field shows the HTTP host name.
- Set the HTTP host name by using the following command:

```
setmqweb properties -k httpHost -v hostName
```

where *hostName* specifies the IP address, domain name server (DNS) host name with domain name suffix, or the DNS host name of the server where IBM MQ is installed. Use an asterisk in double quotation marks to specify all available network interfaces. Use the value `localhost` to allow only local connections.
- Unset the HTTP host name by using the following command:

```
setmqweb properties -k httpHost -d
```

Configuring the HTTP and HTTPS ports

By default, the mqweb server that hosts the IBM MQ Console and REST API uses the HTTPS port 9443. The port that is associated with HTTP connections is disabled. You can enable the HTTP port, configure

a different HTTPS port, or disable the HTTP or HTTPS port. You can configure the ports by using the **setmqweb** command.

Before you begin

If you enable the HTTP port, and are using token based authentication, you must enable the same LTPA token to be used for both HTTP and HTTPS connections. For more information, see [“Configuring the LTPA token” on page 920](#).

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  **z/OS** On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  **Multi** On all other operating systems, you must be a [privileged user](#).
-  **Linux** If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the `mqwebuser.xml` file in the IBM MQ Web Server data directory.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where `WLP_user_directory` is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ_OVERRIDE_DATA_PATH** environment variable to the IBM MQ Web Server data directory.



Attention: By default, the mqweb server requires LTPA tokens to be secured for all requests. If the mqweb server is configured to require LTPA tokens to be secured, you cannot complete the following actions when you connect to the HTTP port:

- Log in to the IBM MQ Console.
- Use token based authentication with the REST API.

To allow LTPA tokens to be used by HTTP requests, set the value of the **secureLTPA** property to `false`. For more information, see [“Configuring the LTPA token” on page 920](#).

Procedure

- View the current configuration of the HTTP and HTTPS ports by using the following command:

```
dspmqweb properties -a
```

The `httpPort` field shows the HTTP port, and the `httpsPort` field shows the HTTPS port.
- Enable or configure the HTTP port: by using the following command:
 - Enable or set the HTTP port by using the following command:

```
setmqweb properties -k httpPort -v portNumber
```

where *portNumber* specifies the port that you want to use for HTTP connections. You can disable the port by using a value of -1.

- Reset the HTTP port value to the default value of -1 by using the following command:

```
setmqweb properties -k httpPort -d
```

- Configure the HTTPS port:

- Set the HTTPS port number by using the following command:

```
setmqweb properties -k httpsPort -v portNumber
```

where *portNumber* specifies the port that you want to use for HTTPS connections. You can disable the port by using a value of -1.

- Reset the HTTPS port number to the default value of 9443 by using the following command:

```
setmqweb properties -k httpsPort -d
```

Configuring the response timeout

By default, the IBM MQ Console and REST API times out if the time taken to send a response back to a client is longer than 30 seconds. You can configure the IBM MQ Console and REST API to use a different timeout value by using the **setmqweb** command.

Before you begin

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  **z/OS** On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  **Multi** On all other operating systems, you must be a [privileged user](#).
-  **Linux** If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the `mqwebuser.xml` file in the IBM MQ Web Server data directory.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ_OVERRIDE_DATA_PATH** environment variable to the IBM MQ Web Server data directory.

Procedure

- View the current configuration of the request timeout by using the following command:

```
dspmqweb properties -a
```

The `mqRestRequestTimeout` field shows the current value for the response timeout. For more information, see [dspmqweb properties](#).
- Set the request timeout by using the following command:

```
setmqweb properties -k mqRestRequestTimeout -v timeout
```

where *timeout* specifies the time, in seconds, before the time out.
- Reset the request timeout to the default value of 30 seconds by using the following command:

```
setmqweb properties -k mqRestRequestTimeout -d
```

Configuring autostart

By default, the IBM MQ Console is automatically started when the mqweb server starts. You can configure whether the IBM MQ Console and the REST API start automatically by using the **setmqweb** command.

Before you begin

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  **z/OS** On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  **Multi** On all other operating systems, you must be a [privileged user](#).
-  **Linux** If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the `mqwebuser.xml` file in the IBM MQ Web Server data directory.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ_OVERRIDE_DATA_PATH** environment variable to the IBM MQ Web Server data directory.

Procedure

- View the current configuration of the autostart by using the following command:

```
dspmqweb properties -a
```


The `mqRestAutostart` field shows whether the REST API is automatically started, and the `mqConsoleAutostart` field shows whether the IBM MQ Console is automatically started.

- Configure whether the IBM MQ Console starts automatically by using the following command:

```
setmqweb properties -k mqConsoleAutostart -v start
```

where *start* is the value `true` if you want the IBM MQ Console to automatically start, or `false` otherwise.

- Configure whether the REST API starts automatically by using the following command:

```
setmqweb properties -k mqRestAutostart -v start
```

where *start* is the value `true` if you want the REST API to automatically start, or `false` otherwise.

Configuring logging

You can configure the logging levels, maximum log file size, and the maximum number of log files that are used by the mqweb server that hosts the IBM MQ Console and REST API. You can configure logging by using the **setmqweb** command.

Before you begin

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  **z/OS** On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  **Multi** On all other operating systems, you must be a [privileged user](#).
-  **Linux** If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the `mqwebuser.xml` file in the IBM MQ Web Server data directory.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ_OVERRIDE_DATA_PATH** environment variable to the IBM MQ Web Server data directory.

About this task

The mqweb server writes log messages and trace to the following log files:

console.log and **messages.log**

These files contain messages that are issued by the IBM MQ Console, the REST API, and the mqweb server that runs these components.

trace.log

This file contains trace for the IBM MQ Console and the REST API. Trace is written to this file only if trace is enabled.

The log files for the mqweb server can be found in one of the following directories:

- In an IBM MQ installation:
 -   On AIX or Linux: `/var/mqm/web/installations/installationName/servers/mqweb/logs`
 -  On Windows:
`MQ_DATA_PATH\web\installations\installationName\servers\mqweb\logs`, where `MQ_DATA_PATH` is the IBM MQ data path. This path is the data path that is selected during the installation of IBM MQ. By default, this path is `C:\ProgramData\IBM\MQ`.
 -  On z/OS: `WLP_user_directory/servers/mqweb/logs`
where `WLP_user_directory` is the directory that was specified when the **crtmqweb** script ran to create the mqweb server definition.
-  In a stand-alone IBM MQ Web Server installation: `MQ_OVERRIDE_DATA_PATH/web/installations/MQWEBINST/servers/mqweb/logs`
where `MQ_OVERRIDE_DATA_PATH` is the IBM MQ Web Server data directory that the **MQ_OVERRIDE_DATA_PATH** environment variable points to.

Messaging trace files for the messaging REST API code that runs in the mqweb server can be found in one of the following directories:

- In an IBM MQ installation:
 -   On AIX or Linux: `/var/mqm/web/installations/installationName/servers/mqweb`
 -  On Windows:
`MQ_DATA_PATH\web\installations\installationName\servers\mqweb`, where `MQ_DATA_PATH` is the IBM MQ data path. This path is the data path that is selected during the installation of IBM MQ. By default, this path is `C:\ProgramData\IBM\MQ`.
 -  On z/OS: `WLP_user_directory/servers/mqweb`
where `WLP_user_directory` is the directory that was specified when the **crtmqweb** script ran to create the mqweb server definition.
-  In a stand-alone IBM MQ Web Server installation: `MQ_OVERRIDE_DATA_PATH/web/installations/MQWEBINST/servers/mqweb`
where `MQ_OVERRIDE_DATA_PATH` is the IBM MQ Web Server data directory that the **MQ_OVERRIDE_DATA_PATH** environment variable points to.

For more information about enabling trace for the:

- REST API, see [Tracing the REST API](#)
- IBM MQ Console, see [Tracing the IBM MQ Console](#)

Procedure

- View the current configuration of the REST API logging by using the following command:
`dspmqweb properties -a`
 - The `maxTraceFileSize` field shows the maximum log file size
 - The `maxTraceFiles` field shows the maximum number of log files

- The `traceSpec` field shows the level of trace that is used
- The `maxMsgTraceFileSize` field shows the maximum messaging trace file size
- The `maxMsgTraceFiles` field shows the maximum number of messaging trace files
- Configure the maximum size of the `messages.log` and `trace.log` files:
 - Set the maximum log file size by using the following command:


```
setmqweb properties -k maxTraceFileSize -v size
```

 where *size* specifies the size, in MB, that each log file can reach.
 - Reset the maximum log file size to the default value of 20 MB by using the following command:


```
setmqweb properties -k maxTraceFileSize -d
```
- Configure the maximum number of the `messages.log` and `trace.log` files:
 - Set the maximum number of each log file by using the following command:


```
setmqweb properties -k maxTraceFiles -v max
```

 where *max* specifies the maximum number of files.
 - Reset the maximum number of each log file to the default value of 2 by using the following command:


```
setmqweb properties -k maxTraceFiles -d
```
- Configure the maximum messaging trace file size:
 - Set the maximum messaging trace file size by using the following command:


```
setmqweb properties -k maxMsgTraceFileSize -v size
```

 where *size* specifies the size, in MB, that each messaging trace file can reach.
 - Reset the maximum messaging trace file size to the default value of 200 MB by using the following command:


```
setmqweb properties -k maxMsgTraceFileSize -d
```
- Configure the maximum number of messaging trace files to use:
 - Set the maximum number of files to use for messaging trace by using the following command:


```
setmqweb properties -k maxMsgTraceFiles -v max
```

 where *max* specifies the maximum number of files.
 - Reset the maximum number of files to use for messaging trace to the default value of 5 by using the following command:


```
setmqweb properties -k maxMsgTraceFiles -d
```
- Configure the level of trace that the mqweb server writes:
 - Set the trace specification that is used by using the following command:


```
setmqweb properties -k traceSpec -v level
```

 where *level* is one of the values that are listed in Table 69 on page 920. The table outlines the logging levels, ordered by increasing level of detail. When you enable a logging level, you also enable each level before it. For example, if you enable the ***=warning** logging level, you also enable ***=severe**, and ***=fatal** logging levels.

 Change this value when requested by IBM Support.
 - Reset the trace specification that is used to the default value of ***=info** by using the following command:


```
setmqweb properties -k traceSpec -d
```

Table 69. Valid logging levels	
Value	Logging level applied
*=off	Logging is turned off.
*=fatal	Task cannot continue and component, application, and server cannot function.
*=severe	Task cannot continue but component, application, and server can still function. This level can also indicate an impending unrecoverable error.
*=warning	Potential error or impending error. This level can also indicate a progressive failure (for example, the potential leaking of resources).
*=audit	Significant event affecting server state or resources
*=info	General information outlining overall task progress
*=config	Configuration change or status
*=detail	General information detailing subtask progress
*=fine	Trace information - General trace + method entry, exit, and return values
*=finer	Trace information - Detailed trace
*=finest	Trace information - A more detailed trace that includes all the detail that is needed to debug problems
*=all	All events are logged

Configuring the LTPA token

LTPA tokens can be used to avoid needing a user to provide username and password credentials on each request to the mqweb server. You can configure the name of the LTPA token cookie, the expiry interval for LTPA authentication tokens, and configure whether LTPA tokens can be used by HTTP connections, by using the **setmqweb** command.

Before you begin

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

- ▶ **z/OS** On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the mqwebuser.xml file.
- ▶ **Multi** On all other operating systems, you must be a [privileged user](#).
- ▶ **Linux** If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the mqwebuser.xml file in the IBM MQ Web Server data directory.

Note: If you are using both the IBM MQ Console, and token authentication with the REST API, the expiry interval is shared.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the WLP_USER_DIR environment variable so that the variable points to your mqweb server configuration.

To set the WLP_USER_DIR environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to **crtmqweb**. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).



Attention: Linux

Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ_OVERRIDE_DATA_PATH** environment variable to the IBM MQ Web Server data directory.

About this task

When users log in to the IBM MQ Console, an LTPA token is generated. If you use token based authentication with the REST API, an LTPA token is generated when the user logs in using the `/login` REST API resource with the HTTP POST method. This token is returned in a cookie. The token is used to authenticate the user without the user being required to log in again with their user ID and password, until the token expires. The default expiry interval is 120 minutes.

The name of the cookie that includes the LTPA token varies by platform:

-  On the IBM MQ Appliance, the LTPA token is `LtpaToken2`. This value cannot be changed.
-   By default, on all other platforms, the name of the cookie that includes the LTPA token starts with `LtpaToken2`, and includes a suffix that can change when the mqweb server is restarted. This randomized cookie name allows more than one mqweb server to run on the same system. However, if you want the cookie name to remain a consistent value, you can specify the name that the cookie has by using the **setmqweb** command.

   If you enable both the HTTP and HTTPS ports, an LTPA token that is issued for an HTTPS request can be reused for an HTTP request. This behavior is disabled by default, but you can enable this behavior by using the **setmqweb** command.

Procedure

- View the current expiry of the LTPA token, the name of the LTPA token cookie, and whether the LTPA token can be used for HTTP requests by using the following command:

```
dspmqweb properties -a
```

 - The `ltpaCookieName` field shows the name of the LTPA token cookie. If you have not set a cookie name, the value of this property is `LtpaToken2_${env.MQWEB_LTPA_SUFFIX}` on AIX, Linux, and Windows, or `LtpaToken2_${httpsPort}` on z/OS. The variable after the `LtpaToken2_` prefix is used by the mqweb server to generate a unique name for the cookie. You cannot set this variable, but you can change the `ltpaCookieName` to a value of your choosing.
 - The `ltpaExpiration` field shows the LTPA token expiry time.
 - The `secureLtpa` field is set to `false` if LTPA tokens can be used by HTTP requests.
- Configure the LTPA token expiry:
 - Set the LTPA token expiry by entering the following command:

```
setmqweb properties -k ltpaExpiration -v time
```

where *time* specifies the time, in minutes, before the LTPA token expires and the user is logged out.

- Reset the LTPA token expiry to the default value of 120 minutes by entering the following command:

```
setmqweb properties -k ltpaExpiration -d
```

-  

Configure the LTPA token cookie name:

- Set the LTPA token cookie name by entering the following command:

```
setmqweb properties -k ltpaCookieName -v name
```

where *name* specifies a unique name for the LTPA token cookie.

- Reset the LTPA token cookie name to the default, where a prefix of `LtpaToken2_` is followed by random characters, by entering the following command:

```
setmqweb properties -k ltpaCookieName -d
```

-  

Configure whether the LTPA token can be used by HTTP connections by entering the following command:

```
setmqweb properties -k secureLtpa -v secure
```

where *secure* specifies whether the LTPA token can be used by both insecure HTTP connections and secure HTTPS connections. A value of `false` allows both HTTP and HTTPS connections to use the same LTPA token.

Configuring remote queue manager connection behavior for the IBM MQ Console

When you use the IBM MQ Console, you can create connections to remote queue managers. That is, you can connect to queue managers that are not part of the same installation as the mqweb server that runs the IBM MQ Console. There are a number of configuration options you can set to control the behavior of the remote queue manager connections.

Before you begin

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  On all other operating systems, you must be a [privileged user](#).
-  If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the `mqwebuser.xml` file in the IBM MQ Web Server data directory.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ_OVERRIDE_DATA_PATH** environment variable to the IBM MQ Web Server data directory.

About this task

You can set the following configuration options:

- Whether remote queue manager connections are allowed.
- Whether the connections can be added by using the IBM MQ Console, or by only the command line.
- Whether local queue managers are displayed in the IBM MQ Console when remote queue manager connections are allowed.
- Whether remote queue manager connections are automatically established when the IBM MQ Console is started, or when there is a connection failure.
- The length of time between each refresh of the list of remote queue managers displayed in the IBM MQ Console.

Procedure

- To view the current remote queue manager connection configuration settings, enter the following command:

```
dspmqweb properties -a
```

 - The `mqConsoleRemoteSupportEnabled` field indicates whether remote queue manager connections are allowed.
 - The `mqConsoleRemoteUIAdmin` field indicates whether remote queue manager connections can be added by using the IBM MQ Console.
 - The `mqConsoleRemoteAllowLocal` field indicates whether the local queue managers are displayed.
 - The `mqConsoleRemotePollTime` field indicates how many seconds there are between each refresh of the list of remote queue managers.
- To prevent or allow remote queue manager connections with the IBM MQ Console, enter the following command:

```
setmqweb properties -k mqConsoleRemoteSupportEnabled -v true or false
```

where *true* allows remote queue manager connections, or *false* prevents remote queue manager connections.

Note:  If the mqweb server runs in a stand-alone IBM MQ Web Server installation, the **mqConsoleRemoteSupportEnabled** property is not valid. The stand-alone IBM MQ Web Server supports connections to only remote queue managers.

- To prevent or allow remote queue manager connections to be added by using the IBM MQ Console, or by only the command line, enter the following command:

```
setmqweb properties -k mqConsoleRemoteUIAdmin -v true or false
```

where `true` allows remote queue manager connections to be added by using the IBM MQ Console and the command line, or `false` allows remote queue manager connections to be added only by using the **setmqweb remote** command on the command line.

- To prevent or allow the display of local queue managers in the IBM MQ Console when remote queue manager connections are allowed, enter the following command:

```
setmqweb properties -k mqConsoleRemoteAllowLocal -v true or false
```

where `true` allows local queue managers to be displayed, or `false` hides the local queue managers.

Note:  If the mqweb server runs in a stand-alone IBM MQ Web Server installation, the **mqConsoleRemoteAllowLocal** property is not valid. The stand-alone IBM MQ Web Server supports connections to only remote queue managers.

- To set the length of time between each refresh of the list of remote queue managers displayed in the IBM MQ Console, enter the following command:

```
setmqweb properties -k mqConsoleRemotePollTime -v seconds
```

where *seconds* is set to an integer value of the number of seconds between each refresh of the list of remote queue managers.

Related reference

 [setmqweb pid](#)
[dspmqweb](#)

Configuring the administrative REST API gateway

When the administrative REST API gateway is enabled, you can perform remote administration with the REST API by using a gateway queue manager. You can configure the queue manager that is used as the default gateway queue manager, or you can prevent remote administration by disabling the administrative REST API gateway, by using the **setmqweb** command.

Before you begin

Note:  If the mqweb server is runs in a stand-alone IBM MQ Web Server installation, this task is not applicable. You cannot use the administrative REST API with the stand-alone IBM MQ Web Server.

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  On all other operating systems, you must be a [privileged user](#).



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

About this task

When the mqweb server runs in an IBM MQ installation, the administrative REST API gateway is enabled by default.

The default gateway queue manager is used when both the following statements are true:

- A queue manager is not specified in the `ibm-mq-rest-gateway-qmgr` header of a REST request.
- The queue manager that is specified in the REST API resource URL is not a local queue manager.

For more information about remote administration with the REST API, see [Remote administration using the REST API](#).

Procedure

- View the current configuration of the administrative REST API gateway by using the following command:

```
dspmweb properties -a
```

The `mqRestGatewayEnabled` field shows whether the gateway is enabled, and `mqRestGatewayQmgr` field shows the name of the default gateway queue manager.

- Configure whether the administrative REST API gateway is enabled by using the following command:

```
setmqweb properties -k mqRestGatewayEnabled -v enabled
```

where *enabled* is the value **true** to enable the administrative REST API gateway, or **false** otherwise.
- Configure which queue manager is used as the default gateway queue manager:

- Set the default gateway queue manager by using the following command:

```
setmqweb properties -k mqRestGatewayQmgr -v qmgrName
```

where *qmgrName* is the name of a queue manager in the same installation as the mqweb server.

- Unset the default gateway queue manager by using the following command:

```
setmqweb properties -k mqRestGatewayQmgr -d
```

Configuring the messaging REST API

You can configure the messaging REST API in a number of ways. You can choose to enable or disable the messaging REST API feature. You can choose the maximum number of pooled connections that can be used by the messaging REST API, and the behavior of the messaging REST API when all connections are in use. You can also choose what user context is used for authorization when you are using the messaging REST API to send, receive, browse, or publish a message.

Procedure

- [“Enabling the messaging REST API” on page 925](#)
- [“Configuring connection pooling for the messaging REST API” on page 927](#)
- [“Configuring the user context that is used for authorization in the messaging REST API” on page 930](#)

Enabling the messaging REST API

You can configure whether the messaging REST API is enabled by using the **setmqweb** command. By default, the messaging REST API is enabled.

Before you begin

To complete this task, you must be a user with certain privileges so that you can use the **dspmweb** and **setmqweb** commands:

- **z/OS** On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the mqwebuser.xml file.
- **Multi** On all other operating systems, you must be a [privileged user](#).
- **Linux** If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the mqwebuser.xml file in the IBM MQ Web Server data directory.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the WLP_USER_DIR environment variable so that the variable points to your mqweb server configuration.

To set the WLP_USER_DIR environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to crtmqweb. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

Procedure

- View the current configuration of the messaging REST API by using the following command:

```
dspmqweb properties -a
```

The mqRestMessagingEnabled field shows whether the messaging REST API is enabled. If the value is True the messaging REST API is enabled.

- Enable the messaging REST API by using the following command:

```
setmqweb properties -k mqRestMessagingEnabled -v true
```

- Disable the messaging REST API by using the following command:

```
setmqweb properties -k mqRestMessagingEnabled -v false
```

Related tasks

[“Configuring connection pooling for the messaging REST API” on page 927](#)

You can configure the maximum number of pooled connections that can be used by the messaging REST API, and the behavior of the messaging REST API when all connections are in use.

[“Configuring the user context that is used for authorization in the messaging REST API” on page 930](#)

You can configure what user context is used for authorization when you are using the messaging REST API to send, receive, browse, or publish a message. That is, you can choose whether the user that is logged in to the messaging REST API, or the user that started the mqweb server, is used for authorization.

[“Configuring the connection mode for the messaging REST API” on page 928](#)

You can configure the messaging REST API to connect to either local or remote queue managers.

Configuring connection pooling for the messaging REST API

You can configure the maximum number of pooled connections that can be used by the messaging REST API, and the behavior of the messaging REST API when all connections are in use.

Before you begin

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  On all other operating systems, you must be a [privileged user](#).
-  If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the `mqwebuser.xml` file in the IBM MQ Web Server data directory.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

Note: Connection pooling is only applicable to the messaging REST API. The administrative REST API and IBM MQ Console/IBM MQ Explorer do not utilize this feature.

About this task

To optimize the performance of the messaging REST API, connections to IBM MQ queue managers are pooled. That is, instead of each REST request creating, using, and deleting its own connection, each REST request uses a connection from a connection pool. By default, 20 connections are available for each queue manager pool, and you can choose from three options for handling requests when all the connections are in use:

- The messaging REST API can create a new, nonpooled connection to use for the request. This behavior is the default behavior.
- The messaging REST API can return an error.
- The messaging REST API can wait for a pooled connection to become available. This wait is an indefinite wait.

You can change the maximum number of pooled connections and the default behavior of the messaging REST API when all connections are in use by using the **setmqweb properties** command.

Note: The messaging REST API uses JMS to connect from the IBM MQ web server to a queue manager. Each pooled connection used for the REST API requires three internal IBM MQ connections to the queue manager, one IBM MQ connection for a JMS Connection and two IBM MQ connections for JMS Sessions. This means that the maximum number of IBM MQ connections that could be established will be three times the maximum pool size that you have configured with **mqRestMessagingMaxPoolSize**.

Procedure

- View the current configuration by using the following command:

```
dspmweb properties -a
```

- The `mqRestMessagingFullPoolBehavior` field shows the behavior of the messaging REST API when all the connections within the pool are in use. If the value is `block`, the messaging REST API must wait for a connection to become available. If the value is `error`, the messaging REST API must return an error. If the value is `overflow`, the messaging REST API must create a nonpooled connection to use, and dispose of the connection after use.
- The `mqRestMessagingMaxPoolSize` field shows the maximum size of the connection pool.
- Configure the behavior of the messaging REST API when all the connections within the pool are in use by using the following command:

```
setmqweb properties -k mqRestMessagingFullPoolBehavior -v action
```

where *action* specifies the action to take. *action* can be one of the following values:

block

When all the connections in the pool are in use, wait for a connection to become available.

error

When all the connections in the pool are in use, return an error.

overflow

When all the connections in the pool are in use, create a nonpooled connection to use, and dispose of the connection after it is used.

- Configure the maximum connection pool size for each queue manager pool by using the following command:

```
setmqweb properties -k mqRestMessagingMaxPoolSize -v size
```

where *size* specifies the size of the pool.

Note: If a large value for `mqRestMessagingMaxPoolSize` is set, and many queue managers are connected to, consider increasing the maximum size of the mqweb server heap. For more information, see [tuning the mqweb server JVM](#).

Related tasks

[“Enabling the messaging REST API” on page 925](#)

You can configure whether the messaging REST API is enabled by using the **setmqweb** command. By default, the messaging REST API is enabled.

[“Configuring the user context that is used for authorization in the messaging REST API” on page 930](#)

You can configure what user context is used for authorization when you are using the messaging REST API to send, receive, browse, or publish a message. That is, you can choose whether the user that is logged in to the messaging REST API, or the user that started the mqweb server, is used for authorization.

[“Configuring the connection mode for the messaging REST API” on page 928](#)

You can configure the messaging REST API to connect to either local or remote queue managers.

Configuring the connection mode for the messaging REST API

You can configure the messaging REST API to connect to either local or remote queue managers.

Before you begin

Note:  If the mqweb server runs in a stand-alone IBM MQ Web Server installation, this task is not applicable. The stand-alone IBM MQ Web Server supports connections to only remote queue managers.

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  **z/OS** On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  **Multi** On all other operating systems, you must be a privileged user.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

About this task

The default connection mode for the messaging REST API varies depending on the type of installation that runs the mqweb server:

- In an IBM MQ installation, by default the messaging REST API connects only to local queue managers in the same installation as the mqweb server. Complete the steps in this task to view and change the connection configuration.
-  **Linux** In a stand-alone IBM MQ Web Server installation, the messaging REST API supports connections to only remote queue managers. The connection configuration cannot be displayed or changed.

Procedure

- View the current configuration of the messaging REST API by using the following command:

```
dspmqweb properties -a
```

The `mqRestMessagingConnectionMode` field shows the current connection mode. If the value is `local`, the messaging REST API can connect only to queue managers in the same installation as the mqweb server. If the value is `remote`, the messaging REST API can connect to remote queue managers.

- Configure the mqweb server to allow the messaging REST API to connect only to queue managers that are in the same installation as the mqweb server by using the following commands:

```
setmqweb properties -k mqRestMessagingConnectionMode -v local  
endmqweb  
strmqweb
```

- Configure the mqweb server to allow the messaging REST API to connect to remote queue managers by using the following command:

```
setmqweb properties -k mqRestMessagingConnectionMode -v remote  
endmqweb  
strmqweb
```

What to do next

If you configure the mqweb server to allow the messaging REST API to connect to remote queue managers, you must provide connection information for each queue manager that you want to connect to. For more information about how to provide the connection information, see [Setting up a remote queue manager for use with the messaging REST API](#).

Related tasks

[“Enabling the messaging REST API” on page 925](#)

You can configure whether the messaging REST API is enabled by using the **setmqweb** command. By default, the messaging REST API is enabled.

[“Configuring connection pooling for the messaging REST API” on page 927](#)

You can configure the maximum number of pooled connections that can be used by the messaging REST API, and the behavior of the messaging REST API when all connections are in use.

[“Configuring the user context that is used for authorization in the messaging REST API” on page 930](#)

You can configure what user context is used for authorization when you are using the messaging REST API to send, receive, browse, or publish a message. That is, you can choose whether the user that is logged in to the messaging REST API, or the user that started the mqweb server, is used for authorization.

Configuring the user context that is used for authorization in the messaging REST API

You can configure what user context is used for authorization when you are using the messaging REST API to send, receive, browse, or publish a message. That is, you can choose whether the user that is logged in to the messaging REST API, or the user that started the mqweb server, is used for authorization.

Before you begin

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  On all other operating systems, you must be a [privileged user](#).
-  If the mqweb server is part of a stand-alone IBM MQ Web Server installation, you must have write access to the `mqwebuser.xml` file in the IBM MQ Web Server data directory.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

About this task

- When the user ID that is used is the user ID that is logged in to the messaging REST API, the **MQMD.UserIDentifier** is set to the user ID that is logged in to the REST API. The **MQMD.AppIdentityData** is set to the user ID that is logged in to the REST API.

- When the user ID that is used is the user ID that started the mqweb server, the **MQMD.UserIDentifier** is left blank. The **MQMD.AppIdentityData** is set to the user ID that is logged in to the REST API.

See [MQMD](#) for more information on the message descriptor parts of the IBM MQ message.

Procedure

- View the current configuration of the messaging REST API by using the following command:

```
dspmqweb properties -a
```

The `mqRestMessagingAdoptWebUserContext` field shows what user ID is used for authorization when you send, publish, receive, or browse messages. If the value is `True`, the user that is logged in to the messaging REST API is used for authorization. If the value is `False` the user that started the mqweb server is used for authorization.

- Configure the messaging REST API to use the user ID of the user that is logged in to the messaging REST API for authorization by using the following command:

```
setmqweb properties -k mqRestMessagingAdoptWebUserContext -v true
```

When `mqRestMessagingAdoptWebUserContext` is set to `true`, the **MQMD.UserIDentifier** is set to the user ID that is logged in to the REST API. The **MQMD.AppIdentityData** is set to the user ID that is logged in to the REST API.

- Configure the messaging REST API to use the user ID of the user that started the mqweb server by using the following command:

```
setmqweb properties -k mqRestMessagingAdoptWebUserContext -v false
```

When `mqRestMessagingAdoptWebUserContext` is set to `false`, the **MQMD.UserIDentifier** is left blank. The **MQMD.AppIdentityData** is set to the user ID that is logged in to the REST API.

Related tasks

[“Enabling the messaging REST API” on page 925](#)

You can configure whether the messaging REST API is enabled by using the **setmqweb** command. By default, the messaging REST API is enabled.

[“Configuring connection pooling for the messaging REST API” on page 927](#)

You can configure the maximum number of pooled connections that can be used by the messaging REST API, and the behavior of the messaging REST API when all connections are in use.

[“Configuring the connection mode for the messaging REST API” on page 928](#)

You can configure the messaging REST API to connect to either local or remote queue managers.

Configuring the REST API for MFT

By default, the REST API for MFT is not enabled. You can configure whether the REST API for MFT is enabled, set the coordination queue manager, set the command queue manager, and specify the MFT reconnect timeout by using the **setmqweb properties** command.

Procedure

- [“Enabling the REST API for MFT” on page 932](#)
- [“Configuring the coordination queue manager for the REST API for MFT” on page 933](#)
- [“Configuring the command queue manager for the REST API for MFT” on page 934](#)
- [“Configuring the REST API for MFT timeout values” on page 935](#)

Enabling the REST API for MFT

Before you can use the REST API for MFT, you must first enable the REST API for MFT. You can configure whether the REST API for MFT is enabled by using the **setmqweb** command. By default, the REST API for MFT is not enabled.

Before you begin

Note:  If the mqweb server runs in a stand-alone IBM MQ Web Server installation, this task is not applicable. You cannot use the REST API for MFT with the stand-alone IBM MQ Web Server.

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the mqwebuser.xml file.
-  On all other operating systems, you must be a [privileged user](#).



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the WLP_USER_DIR environment variable so that the variable points to your mqweb server configuration.

To set the WLP_USER_DIR environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to crtmqweb. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

Procedure

1. View the current configuration of the REST API for MFT by using the following command:

```
dspmqweb properties -a
```

The mqRestMftEnabled field shows whether the REST API for MFT is enabled. The value is True if the REST API for MFT is enabled, or False otherwise.

2. Enable or disable the REST API for MFT by using one of the following commands:

- Enable the REST API for MFT by using the following command:

```
setmqweb properties -k mqRestMftEnabled -v true
```

- Disable the REST API for MFT by using the following command:

```
setmqweb properties -k mqRestMftEnabled -v false
```

3. Restart the mqweb server by entering the following commands:

```
endmqweb  
strmqweb
```


What to do next

If you enabled the REST API for MFT, you must set the coordination queue manager name before you can use the REST API for MFT. For more information about how to set the coordination queue manager, see [“Configuring the coordination queue manager for the REST API for MFT” on page 933](#).

Configuring the coordination queue manager for the REST API for MFT

Before you can use the REST API for MFT, you must configure a queue manager to act as the coordination queue manager for the MFT transactions. You can set which queue manager is the coordination queue manager by using the **setmqweb** command.

Before you begin

Note:  If the mqweb server runs in a stand-alone IBM MQ Web Server installation, this task is not applicable. You cannot use the REST API for MFT with the stand-alone IBM MQ Web Server.

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  On all other operating systems, you must be a privileged user.



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

Procedure

1. View the current configuration of the REST API for MFT by using the following command:

```
dspmqweb properties -a
```

The `mqRestMftCoordinationQmgr` field shows the name of the coordination queue manager.

2. Configure the coordination queue manager by using the following command:

```
setmqweb properties -k mqRestMftCoordinationQmgr -v qmgrName
```

where *qmgrName* is the name of the coordination queue manager. The coordination queue manager must be on the machine where the mqweb server is running. By default, this queue manager name is blank. If a value is not set, the REST API for MFT does not work.

3. Restart the mqweb server by entering the following commands:

```
endmqweb  
strmqweb
```

What to do next

- Ensure that the REST API for MFT is enabled. For more information, see [“Enabling the REST API for MFT” on page 932](#).
- If you want to use the REST API for MFT to submit create requests, you must set the command queue manager name. For example, if you want to use a REST API command such as **create transfer**, you must set the command queue manager name. For more information, see [“Configuring the command queue manager for the REST API for MFT” on page 934](#).
- You can configure the REST API for MFT timeout values. The default timeout is 30 minutes. For more information, see [“Configuring the REST API for MFT timeout values” on page 935](#).
- To use the REST API for MFT, a user must be authenticated to the mqweb server and must be a member of one or more of the MFTWebAdmin, or MFTWebAdminRO roles. For more information about configuring users, see [Configuring users and roles for the REST API](#).

Configuring the command queue manager for the REST API for MFT

Before you can use the REST API for MFT to submit create requests, you must set the command queue manager name. For example, to use the **create transfer** resource, you must set the command queue manager name. You can set the command queue manager name by using the **setmqweb** command.

Before you begin

Note:  If the mqweb server runs in a stand-alone IBM MQ Web Server installation, this task is not applicable. You cannot use the REST API for MFT with the stand-alone IBM MQ Web Server.

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the `mqwebuser.xml` file.
-  On all other operating systems, you must be a [privileged user](#).



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the `WLP_USER_DIR` environment variable so that the variable points to your mqweb server configuration.

To set the `WLP_USER_DIR` environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where `WLP_user_directory` is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

Procedure

1. View the current configuration of the REST API for MFT by using the following command:

```
dspmqweb properties -a
```

The `mqRestMftCommandQmgr` field shows the name of the command queue manager.

2. Configure the command queue manager by using the following command:

```
setmqweb properties -k mqRestMftCommandQmgr -v qmgrName
```

where *qmgrName* is the name of the command queue manager. The command queue manager must be on the machine where the mqweb server is running. By default, this queue manager name is blank. If a value is not set, the REST API for MFT for a create command does not work.

- Restart the mqweb server by entering the following commands:

```
endmqweb  
strmqweb
```

What to do next

- Ensure that the REST API for MFT is enabled. For more information, see [“Enabling the REST API for MFT”](#) on page 932.
- Ensure that a coordination queue manager is set. For more information, see [“Configuring the coordination queue manager for the REST API for MFT”](#) on page 933.
- You can configure the REST API for MFT timeout values. The default timeout is 30 minutes. For more information, see [“Configuring the REST API for MFT timeout values”](#) on page 935.
- To use the REST API for MFT, a user must be authenticated to the mqweb server and must be a member of one or more of the MFTWebAdmin, or MFTWebAdminRO roles. For more information about configuring users, see [Configuring users and roles for the REST API](#).

Configuring the REST API for MFT timeout values

You can configure the length of time, in minutes, after which the REST API for MFT stops trying to connect to the coordination queue manager after the connection is broken. The default timeout is 30 minutes. You can configure this timeout by using the **setmqweb** command.

Before you begin

Note:  If the mqweb server is runs in a stand-alone IBM MQ Web Server installation, this task is not applicable. You cannot use the REST API for MFT with the stand-alone IBM MQ Web Server.

To complete this task, you must be a user with certain privileges so that you can use the **dspmqweb** and **setmqweb** commands:

-  On z/OS, you must have authority to run the **dspmqweb** and **setmqweb** commands, and write access to the mqwebuser.xml file.
-  On all other operating systems, you must be a [privileged user](#).



Attention:

Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the WLP_USER_DIR environment variable so that the variable points to your mqweb server configuration.

To set the WLP_USER_DIR environment variable, issue the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to crtmqweb. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see [Create the mqweb server](#).

About this task

You can configure the timeout for the REST API for MFT.

The REST API for MFT attempts to re-establish the connection immediately after the connection to the coordination queue manager is broken. If this attempt fails, there is an interval of five minutes between every reconnection attempt, until the timeout has passed. Therefore, setting a value between 0-5 results in only one attempt to reconnect.

After the reconnection times out, the next attempt to reconnect is made when any of the REST API for MFT resources are invoked. If this reconnection attempt fails, MFT again attempts to reconnect every five minutes until the reconnect timeout has passed.

Procedure

1. View the current configuration of the REST API for MFT by using the following command:

```
dspmweb properties -a
```

The `mqRestMftReconnectTimeoutInMinutes` field shows the reconnection timeout value, until the MFT Transfer Rest services stops attempting to connect to the coordination queue manager.

2. Configure the timeout, in minutes, after which the REST API for MFT stops trying to connect to the coordination queue manager:

- Reset the timeout to the default value of 30 minutes:

```
setmqweb properties -k mqRestMftReconnectTimeoutInMinutes -d
```

- Set the timeout:

```
setmqweb properties -k mqRestMftReconnectTimeoutInMinutes -v time
```

where *time* specifies the time, in minutes, before the timeout occurs.

If this value is set between 0-5, the REST API for MFT tries to reconnect to the coordination queue manager only once. If the connection fails, there are no attempts to re-establish the connection until the REST API is invoked.

If this value is set to -1, the REST API for MFT attempts to reconnect until the connection is successful.

3. Restart the mqweb server by entering the following commands:

```
endmqweb  
strmqweb
```

What to do next

- Ensure that the REST API for MFT is enabled. For more information, see [“Enabling the REST API for MFT” on page 932](#).
- Ensure that a coordination queue manager is set. For more information, see [“Configuring the coordination queue manager for the REST API for MFT” on page 933](#).
- If you want to use the REST API for MFT to submit create requests, you must set the command queue manager name. For example, if you want to use a REST API command such as **create transfer**, you must set the command queue manager name. For more information, see [“Configuring the command queue manager for the REST API for MFT” on page 934](#).
- To use the REST API for MFT, a user must be authenticated to the mqweb server and must be a member of one or more of the MFTWebAdmin, or MFTWebAdminRO roles. For more information about configuring users, see [Configuring users and roles for the REST API](#).

Tuning the mqweb server JVM

By default, the mqweb server Java Virtual Machine (JVM) uses platform-specific defaults for configuration parameters such as the minimum and maximum size of the heap, and the size of the class cache.

About this task

You might need to change the default values to improve performance or to resolve problems. For example, if a `java.lang.OutOfMemoryError` is thrown by the mqweb server, you must increase the maximum size of the heap. You should also increase the size of the heap if you are trying to load a large number of queue objects.

If you are experiencing issues with the display of dashboard configuration information in the IBM MQ Console, you must set a variable that determines the file encoding of the configuration. You can change the default values in the `jvm.options` file.

Procedure

1. Open the `jvm.options` file.

The `jvm.options` file can be found in one of the following directories:

- In an IBM MQ installation:

-   On AIX or Linux: `/var/mqm/web/installations/installationName/servers/mqweb`
-  On Windows:
`MQ_DATA_PATH\web\installations\installationName\servers\mqweb`, where `MQ_DATA_PATH` is the IBM MQ data path. This path is the data path that is selected during the installation of IBM MQ. By default, this path is `C:\ProgramData\IBM\MQ`.
-  On IBM i: `MQ_DATA_PATH/web/installations/Installation1/`
-  On z/OS: `WLP_user_directory/servers/mqweb`
where `WLP_user_directory` is the directory that was specified when the **crtmqweb** script ran to create the mqweb server definition.

-  In a stand-alone IBM MQ Web Server installation: `MQ_OVERRIDE_DATA_PATH/web/installations/MQWEBINST/servers/mqweb`
where `MQ_OVERRIDE_DATA_PATH` is the IBM MQ Web Server data directory that the **MQ_OVERRIDE_DATA_PATH** environment variable points to.

2. Optional: Set the maximum heap size by adding the following line to the file:

```
-XmxMaxSizeM
```

Where *MaxSize* specifies the maximum size of the heap, in MB.

For example, the following line sets the maximum heap size to 1GB:

```
-Xmx1024m
```

3. Optional: Set the minimum heap size by adding the following line to the file:

```
-XmsMinSizeM
```

Where *MinSize* specifies the minimum size of the heap, in MB. Increasing the minimum heap size from the default can reduce the time it takes to start the mqweb server.

For example, the following line sets the minimum heap size to 512MB:

```
-Xms512m
```

4. Optional: Set the size of the class cache by adding the following line to the file:

```
-XscmxSizeM
```

Where *Size* specifies the class cache size, in MB.

For example, the following line sets the class cache size to 100MB:

```
-Xscmx100m
```

The Java shared class cache is used to store data such as loaded classes and Ahead-Of-Time (AOT) compiled code.

The class cache significantly reduces the time taken to start the mqweb server. The first time the mqweb server is started the class cache is created and the server can take a significant time to start. Subsequent restarts of the server will be much quicker as classes can be loaded from the shared class cache.

Increasing the size of the class cache from the default can reduce the time it takes to start the mqweb server.

 The class cache is recreated when the mqweb server is started on a different z/OS system. Therefore, starting the mqweb server on a different z/OS system in a sysplex can take significantly longer than restarting the server on the same system.

Note that changes to this value take effect only when the class cache is created. The class cache is created when the mqweb server is first started, or after the class cache has been destroyed using the Java class cache utility.

5. Required: Check that the file contains the following lines to specify the file encoding that is used when the REST API processes data, and for the user dashboard configuration information in the IBM MQ Console:

```
-Dfile.encoding=UTF-8  
-Ddefault.client.encoding=UTF-8
```

6. Restart the mqweb server.

 On z/OS, stop and restart the mqweb server started task.

 On all other platforms, enter the following commands on the command line:

```
endmqweb  
strmqweb
```

File structure of the IBM MQ Console and REST API installation component

There are two sets of directory structures that are associated with the IBM MQ Console and REST API installation component. One directory structure contains files that can be edited. The other directory structure contains files that cannot be edited.

Editable files

The user editable files are laid down as part of the initial installation of the IBM MQ Console and REST API installation component. As these files can be edited, the files are not changed when maintenance is applied.

The location of the user editable files depends on the operating system and the product that is installed.

- In an IBM MQ installation, the user editable files are in one of the following directories:

–   On AIX or Linux: /var/mqm/web/installations/
installationName

- **Windows** On Windows: `MQ_DATA_PATH\web\installations\installationName`, where `MQ_DATA_PATH` is the IBM MQ data path. This path is the data path that is selected during the installation of IBM MQ. By default, this path is `C:\ProgramData\IBM\MQ`.
- **z/OS** On z/OS: the directory that was specified when the `crtmqweb` script ran to create the mqweb server definition.
- **Linux** In a stand-alone IBM MQ Web Server installation: `MQ_OVERRIDE_DATA_PATH/web/installations/MQWEBINST`
 where `MQ_OVERRIDE_DATA_PATH` is the IBM MQ Web Server data directory that the **`MQ_OVERRIDE_DATA_PATH`** environment variable points to.

Under this top-level directory, the following directories and files are present:

Directories and files	Description
<code>angular.persistence/</code>	Directory where the IBM MQ Console dashboard configuration is stored.
<code>servers/</code>	WebSphere Liberty servers directory.
<code>servers/mqweb</code>	Directory that contains the mqweb server directory structure.
<code>servers/mqweb/logs</code>	Directory that contains logs for the mqweb server.
<code>servers/mqweb/logs/console.log</code>	Log of basic server status and operation messages.
<code>servers/mqweb/logs/ffdc</code>	First Failure Data Capture (FFDC) output directory.
<code>servers/mqweb/logs/messages.log</code>	Log of runtime messages from the mqweb server, including the IBM MQ Console and REST API. Older messages are stored in files that are called <code>messages_timestamp.log</code> .
<code>servers/mqweb/logs/trace.log</code>	Log of trace from the mqweb server, including the IBM MQ Console and REST API. Older trace is stored in files that are called <code>trace_timestamp.log</code> . These files exist only if trace is enabled.
<code>servers/mqweb/logs/state</code>	Server-specific state.
<code>servers/mqweb/server.xml</code>	Main server configuration file. This file is read only. Edit the <code>mqwebuser.xml</code> file to override the default configuration.
<code>servers/mqweb/mqwebuser.xml</code>	Configuration file for the IBM MQ Console and REST API. Settings that are configured in this file override the default configuration. You must be a privileged user to edit this file.
<code>servers/mqweb/resources</code>	Directory that contains various server resources such as keystores.
<code>servers/mqweb/workarea</code>	Directory that is created by the server as it operates. This directory is created after the server is first run.

Non-editable files

The non-editable files are laid down as part of the initial installation of the IBM MQ Console and REST API installation component. These files are updated when maintenance is applied.

The location of the non-editable files depends on the operating system and the product that is installed.

- In an IBM MQ installation, the non-editable files are in one of the following directories:

-  On AIX, Linux, and Windows: `MQ_INSTALLATION_PATH/web`
-  On IBM i: `MQ_INSTALLATION_PATH/web`
-  On z/OS: `installation_directory/web/`

where *installation_directory* is the IBM MQ for z/OS UNIX System Services Components installation path.

-  In a stand-alone IBM MQ Web Server installation, the directory to which the IBM MQ Web Server installation file was decompressed.

The following directory structure and files are present in this location:

Directories and files	Description
bin/	Directory that contains WebSphere Liberty commands. You must be a <u>privileged user</u> to execute scripts in this directory.
mq/	Directory structure that contains various IBM MQ resources.
mq/apps/	Directory that contains the IBM MQ Console and REST API applications.
mq/etc/	
mq/etc/mqweb.xml	Read-only configuration file for the mqweb server. Edit the mqwebuser.xml file to make configuration changes.
mq/libs	Directory that contains shared libraries for use by the IBM MQ Console and REST API.
mq/samp	Directory that contains samples.
mq/samp/configuration	Directory that contains sample configuration files that can be copied into the mqwebuser.xml file.

Backing up and restoring your mqweb server configuration

You can back up your mqweb server configuration, and restore it to the same location, or a different location.

Before you begin

Before you can restore your mqweb server configuration, you must install IBM MQ, or the stand-alone IBM MQ Web Server, on the system where you want to restore the mqweb server. In a stand-alone IBM MQ Web Server installation, you must create the mqweb server by following the steps in [“Configuring the stand-alone IBM MQ Web Server” on page 911](#).

About this task

Follow the procedure in this task to back up and restore your mqweb server configuration. If you restore your mqweb server to a different location, you must update the mqweb server configuration to ensure that references to files are correct.

You can also use this procedure to migrate a mqweb server that currently runs in an IBM MQ installation to run in a stand-alone IBM MQ Web Server installation.

Procedure

1. To back up the mqweb server configuration, copy all the files in the directory that contains the mqweb server configuration to your backup location.
 - In an IBM MQ installation, copy the contents of the following directory:
 -   On AIX or Linux: `/var/mqm/web/installations/installationName`
 -  On Windows: `MQ_DATA_PATH\web\installations\installationName`, where `MQ_DATA_PATH` is the IBM MQ data path. This path is the data path that is selected during the installation of IBM MQ. By default, this path is `C:\ProgramData\IBM\MQ`.
 -  On z/OS: the WebSphere Liberty user directory that was specified when the **crtmqweb** script ran to create the mqweb server definition.
 -  In a stand-alone IBM MQ Web Server installation, copy the contents of the `MQ_OVERRIDE_DATA_PATH/web/installations/MQWEBINST` directory, where `MQ_OVERRIDE_DATA_PATH` is the IBM MQ Web Server data directory that the **MQ_OVERRIDE_DATA_PATH** environment variable points to.
2. To restore the mqweb server configuration, replace the contents of the directory that contains the mqweb server configuration with the files that you copied in step “1” on page 941.
 - In an IBM MQ installation, replace the contents of the following directory:
 -   On AIX or Linux: `/var/mqm/web/installations/installationName`
 -  On Windows: `MQ_DATA_PATH\web\installations\installationName`, where `MQ_DATA_PATH` is the IBM MQ data path. This path is the data path that is selected during the installation of IBM MQ. By default, this path is `C:\ProgramData\IBM\MQ`.
 -  On z/OS: the WebSphere Liberty user directory that was specified when the **crtmqweb** script ran to create the mqweb server definition.
 -  In a stand-alone IBM MQ Web Server installation, replace the contents of the `MQ_OVERRIDE_DATA_PATH/web/installations/MQWEBINST` directory, where `MQ_OVERRIDE_DATA_PATH` is the IBM MQ Web Server data directory that the **MQ_OVERRIDE_DATA_PATH** environment variable points to.
3. Set the ownership of the files that you restored in step “2” on page 941 so that the mqweb server user ID can access the files.
4. If you restored the mqweb server configuration to a different location, change the value of any properties in the mqweb server configuration that refer to files in the previous mqweb server configuration directory.
 - a) Before you issue either the **setmqweb** or **dspmqweb** commands, set your environment to point to your mqweb server configuration.

- **z/OS** On z/OS, set the **WLP_USER_DIR** environment variable so that the variable points to your mqweb server configuration, by entering the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

where *WLP_user_directory* is the name of the directory that is passed to the **crtmqweb** command.

For more information, see [Create the mqweb server](#).

- **Linux** In a stand-alone IBM MQ Web Server installation, set the **MQ_OVERRIDE_DATA_PATH** environment variable to the IBM MQ Web Server data directory.
 - In all other environments, you do not need to complete any actions to set your environment.
- b) View the value of all configurable mqweb server properties that a user has modified. Issue the following command:

```
dspmqweb properties -u
```

- c) If the **remoteKeyfile** property is displayed, check the value of the property.
If the value of the property refers to a file path in the previous mqweb server configuration directory, change the value to refer to the file path in the new mqweb server configuration directory. Issue the following command to change the value of the **remoteKeyfile** property:

```
setmqweb properties -k remoteKeyfile -v path_to_keyfile
```

- d) View the mqweb server remote queue manager configuration. Issue the following command:

```
dspmqweb remote -a
```

- e) If any of the following properties are displayed, check the value of the property:

- **globalTrustStorePath**
- **globalKeyStorePath**
- **ccdtURL**
- **keyStorePath**
- **trustStorePath**

Change the value of any property that refers to a file path in the previous mqweb server configuration directory to refer to the file path in the new mqweb server configuration directory. Issue the **setmqweb remote** command to change the value of each property. For example, to change the value of the **keyStorePath** property for the remote queue manager with the unique name **remote-QM1**, issue the following command:

```
setmqweb remote -uniqueName remote-QM1 -keyStorePath new_keystore_path
```

For more information, see [setmqweb remote \(set mqweb server remote queue manager configuration\)](#).

Windows Linux MQ Adv. MQ Adv. VUE MQ Adv. z/OS Defining an Aspera gateway connection on Linux or Windows platforms

The IBM Aspera faspio Gateway provides a fast TCP/IP tunnel that can significantly increase network throughput for IBM MQ. A queue manager running on any entitled platform can connect through an Aspera gateway. The gateway itself is deployed on Red Hat or Ubuntu Linux, or Windows.

About this task

The Aspera gateway can be used to improve the performance of queue manager channels. It is especially effective if the network has high latency or tends to lose packets, and it is typically used to speed up the connection between queue managers in different data centers.

Note: For a fast network that does not lose packets there is a decrease in performance when using the Aspera gateway, so it is important to check network performance before and after defining an Aspera gateway connection.

You define an Aspera gateway at each end of the IP network connection, then use TCP/IP to connect queue manager channels to each gateway. A queue manager does not need to be running on the same machine as the Aspera gateway that it uses, and multiple queue managers can use the same gateway.

To use the Aspera gateway, you must have one or more of the following entitlements:

-  IBM MQ Advanced for Multiplatforms
-  IBM MQ Appliance
-  IBM MQ Advanced for z/OS VUE
-  IBM MQ Advanced for z/OS

You can deploy the Aspera gateway on any of the following platforms:

- Linux for x86-64
- Linux on Power® Systems - Little Endian
- Linux on IBM Z®
- Windows - for more information about platform support on Windows, see the [IBM Aspera faspio Gateway documentation](#).

Use of the Aspera gateway is limited to IBM MQ messages unless the gateway is separately entitled.

Queue managers that use the Aspera gateway can be running on any supported platform. For a complete list of supported platforms, see [Icons used in the product documentation](#).

For each queue manager that is not on the same machine as the Aspera gateway that it uses, check that you have a fast network connection between the queue manager and the Aspera gateway.

You use a `toml` file to create a gateway definition that defines the inbound and outbound ports that the gateway uses. A sample `toml` file is shipped with the Aspera gateway. The outbound gateway definition defines the connection from the local queue manager to the gateway, and from the local gateway to the remote gateway. The inbound gateway definition defines the connection from the remote gateway to the local gateway, and from the local gateway to the local queue manager.

The following steps provide a basic guide to getting up and running. For more detailed information, see the [IBM Aspera faspio Gateway documentation](#).

Procedure

1. Get the Aspera gateway installation image.

 For Multiplatforms, you download the Aspera gateway from Passport Advantage. The download is labeled "IBM Aspera faspio Continuous Delivery Release for IBM MQ V10.0 Multiplatform Multilingual eAssembly". It is delivered as a Continuous Delivery (CD) image only because of the pace of change in this area, which means that updates are needed at the frequency of the CD releases, and you can install it on any IBM MQ system that has IBM MQ Advanced for Multiplatforms or IBM MQ Appliance entitlement. To download this eAssembly, go to [Downloading IBM MQ 10](#) then click the tab for the required release. The eAssembly contains installation images for all platforms on which the gateway is available. The eAssembly also contains a `ibm-faspio-license.zip` file, which contains a license file.

  If your IBM MQ system has IBM MQ Advanced for z/OS VUE entitlement, or IBM MQ Advanced for z/OS entitlement, you get the Aspera gateway from the Connector Pack component that is part of the SMP/E installation.

MQ Adv. VUE **MQ Adv. z/OS** The files for IBM MQ Advanced for z/OS VUE and IBM MQ Advanced for z/OS are as follows:

Table 70. File names and faspio version numbers by platform and IBM MQ version		
Platform	File name	faspio version number
Linux for x86-64	 M11QQEN	1.3.7
Linux on Power Systems - Little Endian	 M11QREN	1.3.7
Linux on IBM Z	 M11QSEN	1.3.7
Windows	 M11QTEN	1.3.7

Notes:

- a. The Aspera gateway cannot run natively on z/OS.

MQ Adv. VUE **MQ Adv. z/OS** In addition to the installation images, the fasp directory contains M05QKEN.zip, which contains a license file.

2. Copy the Aspera gateway installation image onto the two machines that will run the gateway, then extract and install the gateway.

Use the license file contained in ibm-faspio-license.zip (Multiplatforms) or M05QKEN.zip (z/OS). For more information, see the IBM Aspera faspio Gateway documentation:

-  [Installing on Linux](#)
-  [Installing on Windows](#)

3. Configure and secure each gateway.

For more information, see the IBM Aspera faspio Gateway documentation:

- [Configuring the Gateway configuration file](#)
- [Securing the Gateway](#)

4. At each end of the network connection, change the channel definition to connect to the port that the local gateway is listening on.

5. Start each gateway service.

For more information, see the IBM Aspera faspio Gateway documentation:

-  [Launching on Linux](#)
-  [Launching on Windows](#)

6. Restart the channels.

Your queue managers are now communicating across an Aspera gateway connection.

Example

This example defines an Aspera gateway connection on two machines running Linux. The configuration is as follows:

- The IP address of the local gateway machine is 9.20.193.107. The IP address of the remote gateway machine is 9.20.192.115.

- The local queue manager is running on a machine with IP address 9.20.121.5. The remote queue manager is running on a machine with IP address 9.20.121.25. Both queue managers are listening on port 1414.
- The queue manager channel on the local queue manager is changed to connect to the local Aspera gateway using **conname** 9.20.193.107(1500). The queue manager channel on the remote queue manager is changed to connect to the remote Aspera gateway using **conname** 9.20.192.115(1500).
- From IBM Aspera faspio Gateway 1.2, TLS is enabled by default. If you want to configure TLS with the gateway, see [Securing the Gateway](#) in the IBM Aspera faspio Gateway documentation.

1. Define an Aspera gateway connection on the local gateway machine:

- Install the Aspera gateway:

–  On Linux, use the following command:

```
rpm -ivh ibm-faspio-gateway-<version>.x86_64.rpm
```

- Modify the gateway.toml file in the directory that was created by the install:

Edit the file to set the local gateway definitions.

```
[[bridge]]
  name = "Outbound"
  [bridge.local]
    protocol = "tcp"
    host = "9.20.193.107"
    port = 1500
  tls_enabled = false

  [bridge.forward]
    protocol = "fasp"
    host = "9.20.192.115"
    port = 1600
  tls_enabled = false

[[bridge]]
  name = "Inbound"
  [bridge.local]
    protocol = "fasp"
    host = "9.20.193.107"
    port = 1600
  tls_enabled = false

  [bridge.forward]
    protocol = "tcp"
    host = "9.20.121.5"
    port = 1414
  tls_enabled = false
```

- For faspio 1.3.4, copy the aspera-license file from ibm-faspio-license.zip (Multiplatforms) or M05QKEN.zip (z/OS) into /usr/local/etc/faspio/.

For Faspio 1.3.5, copy the aspera-license file from ibm-faspio-license.zip (Multiplatforms) or M05QKEN.zip (z/OS) into /etc/faspio-gateway.

2. Repeat the previous step to define an Aspera gateway connection on the remote gateway machine.

- Modify the gateway.toml file in the directory that was created by the install. Edit the file to set the remote gateway definitions:

```
[[bridge]]
  name = "Outbound"
  [bridge.local]
    protocol = "tcp"
    host = "9.20.193.107"
    port = 1500
  tls_enabled = false

  [bridge.forward]
    protocol = "fasp"
    host = "9.20.192.115"
    port = 1600
  tls_enabled = false
```

```
[[bridge]]
  name = "Inbound"
  [bridge.local]
    protocol = "fasp"
    host = "9.20.193.107"
    port = 1600
  tls_enabled = false

  [bridge.forward]
    protocol = "tcp"
    host = "9.20.121.5"
    port = 1414
  tls_enabled = false
```

- For faspio 1.3.4 copy the aspera-license file from either `ibm-faspio-license.zip` (Multiplatforms) or `M05QKEN.zip` (z/OS) into `/usr/local/etc/faspio/`.
- For Faspio 1.3.5, copy the aspera-license file from `ibm-faspio-license.zip` (Multiplatforms) or `M05QKEN.zip` (z/OS) into `/etc/faspio-gateway`.
- 3. At each end of the connection, change the channel definition to connect to the port that the local gateway is listening on.
 - Change the queue manager channel on the local queue manager to connect to the local Aspera gateway using **conname** 9.20.193.107(1500).
 - Change the queue manager channel on the remote queue manager to connect to the remote Aspera gateway using **conname** 9.20.192.115(1500).
- 4. Start the local gateway by running the following command on the local gateway machine:
 - 

```
sudo systemctl start faspio-gateway
```
- 5. Start the remote gateway by running the following command on the remote gateway machine:
 - 

```
sudo systemctl start faspio-gateway
```
- 6. Restart the channels.

What to do next

The Aspera gateway passes on the data it receives, without interpreting it in any way. This means that you can configure TLS between the queue manager channels that are using the Aspera gateway because the gateway connection is unaware of the TLS handshaking. This also means that queue managers on any supported IBM MQ platform can use the Aspera gateway.

To use a multi-instance queue manager with the gateway, configure gateway definitions for each instance of the queue manager.

Note: The Aspera gateway has only been tested with queue manager channels. It has not been tested with client channels. This is because the envisaged use for the Aspera gateway is to connect remote queue managers over a slow network, whereas client applications typically connect to queue managers in a local data center over a fast network.

Related reference

“Which communication type to use” on page 17

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

[IBM Aspera faspio Gateway documentation](#)

Multi **Configuring IBM MQ for use with the IBM Cloud Private metering service**

Configuring IBM MQ for use with the IBM Cloud Private metering service to report and view queue manager startup and usage information.

Before you begin

Before you configure your IBM MQ queue managers to use an IBM Cloud Private service, you must have an IBM Cloud account. To create your account, see [Sign up for IBM Cloud](#).

About this task

By using the IBM Cloud Private metering service, you can connect your on-premises IBM products to your service instance in IBM Cloud Private and see all the registered products in your organization in a single dashboard.

You can configure and connect your AIX, Linux, and Windows queue managers to your metering service instance and see their startup and usage information. However, on platforms other than Linux Container environments, the data cannot be used in support of Hourly Container-Based pricing licenses.

To record usage data for a monthly VPC license type, instead of the default hourly licensing metric, set the environment variable `AMQ_LICENSING_METRIC=VPCMonthlyPeak`. This causes the queue manager to upload data relating to monthly VPC license types, instead of the default behavior of uploading data related to hourly container-based licenses.

Use the following attributes with the `ReportingService` stanza in the `qm.ini` file:

APIKeyFile

Location of the text file with the metering service instance **APIKey** value.

CapacityReporting

Writes error log messages periodically to the AMQERR logs in the following format:

```
4/22/2020 01:44:29 PM - Process(1274.1) User(bld-adm) Program(amqmgr0)
Host(8b3b83f2bc7d) Installation(Docker)
VRMF(9.2.0.0)
Time(2020-04-22T13:44:29.295Z)
ArithInsert1(300)
CommentInsert1(8.5)
CommentInsert2(IBM MQ Advanced)
```

The information produced by the **CapacityReporting** attribute is inserted into message AMQ5064, which gives you a better understanding of how much IBM MQ your enterprise is using:

AMQ5064

This queue manager has been running for 300 seconds. It is currently running with 8.5 cores.
License type is IBM MQ Advanced.

Severity

0: Information

Explanation

This is an information message for usage tracking.

Response

None.

LicensingGroup

The billing group that the queue manager belongs to. This affects the way data is grouped in reports generated by the metering service.

ServiceURL

The IBM Cloud Private service address.

ServiceProxy

The URL and port for the HTTP proxy that can be used if the queue managers do not have direct access to the network that the metering service is running on.

You can see the hosts that your products are installed on, the product versions that you are using and the platforms that they are running on. From the high-level usage metrics that are displayed for each product, you can have an overview of how heavy the workloads are. For IBM MQ, you can see which queue managers are more heavily used and which ones have lighter workloads.

When a queue manager is configured to connect to an instance of the metering service, the following information is reported to IBM Cloud Private:

- IBM MQ queue manager name
- IBM MQ queue manager identifier
- IBM MQ installation root directory
- IBM MQ installed components (name and version)
- Host name
- Host operating system name
- Host operating system version
- Virtual Processor Core (VPC) usage information for the IBM MQ queue manager

You can monitor your queue manager VPC usage metrics in your metering service instance dashboard.

Procedure

- Configure a queue manager for use with the metering service instance on IBM Cloud Private.
- Connect to the IBM Cloud Private metering service through an HTTP proxy.
- Troubleshoot the connection to the IBM Cloud Private metering service.

Related reference

[Pricing metric for Virtual Processor Cores \(VPCs\)](#)

Configuring a queue manager for use with the metering service instance on IBM Cloud Private

Set up security and IBM Cloud registration information for your queue manager, and then connect to the metering service instance that you already created.

About this task

Your [IBM Cloud Private metering service](#) instance dashboard shows data for only the queue managers that are configured to include the security and IBM Cloud Private registration information.

Procedure

1. Follow the IBM Cloud Private documented steps for creating a service ID at:
[Creating a service ID by using IBM Cloud Private CLI.](#)
2. Follow the IBM Cloud Private documented steps for creating an API key at:
[API key management APIs.](#)
3. Download the TLS certificates from the IBM Cloud Private cluster.
Make a note of the location where you downloaded the certificates to. You can add the downloaded certificates to the key repository for your queue manager, in step “9” on [page 949](#).
4. Create a text file `apikeyfile.txt` and add the **API key** value that you copied in the previous task.
Note the location of the `apikeyfile.txt` so you can include the path to it in [step 8](#). This file must be readable by the queue manager user (`'mqm'` on AIX and Linux systems). The file

must contain only the **API key** itself, not a JSON payload, for example d9c11b45-4dda-4de4-c0b2-2e4e1004dc64.

5. Create the queue manager, for example, *QM1*.

For more information, see [Creating and managing queue managers on Multiplatforms](#).

6. Start the queue manager *QM1*.

For more information, see [Starting a queue manager](#).

7. Remember to set up your IBM MQ command line environment before you run IBM MQ commands.

Run the **setmqenv** command.

AIX On AIX:

```
. /usr/mqm/bin/setmqenv -s
```

Linux On Linux:

```
. /opt/mqm/bin/setmqenv -s
```

Windows On Windows:

```
"C:\Program Files\IBM\MQ\bin\setmqenv.cmd" -n installation name
```

8. Create an SSL truststore for the queue manager *QM1*.

AIX Begin creating the truststore, on AIX:

```
runmqakm -keydb -create -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms  
-expire 30 -stash
```

Linux On Linux:

```
runmqakm -keydb -create -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms  
-expire 30 -stash
```

Windows On Windows:

```
runmqakm -keydb -create -db "MQ data directory\qmgrs\QM1\ssl\key.kdb" -pw password -type  
cms -expire 30 -stash
```

9. Add the digital certificates that you downloaded in step “3” on page 948, to the queue manager's key repository.

AIX On AIX:

```
runmqakm -cert -add -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms  
-label RootCA  
-file Download_location/RootCA.crt -format ascii -trust enable  
runmqakm -cert -add -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms  
-label ServerCert  
-file Download_location/CERT.crt -format ascii -trust enable
```

Linux On Linux:

```
runmqakm -cert -add -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms  
-label RootCA  
-file Download_location/RootCA.crt -format ascii -trust enable  
runmqakm -cert -add -db MQ data directory/qmgrs/QM1/ssl/key.kdb -pw password -type cms  
-label ServerCert  
-file Download_location/CERT.crt -format ascii -trust enable
```

Windows On Windows:

```
runmqakm -cert -add -db "MQ data directory\qmgrs\QM1\ssl\key.kdb" -pw password -type cms  
-label RootCA  
-file "Download_location\RootCA.crt" -format ascii -trust enable  
  
runmqakm -cert -add -db "C:\ProgramData\IBM\MQ\qmgrs\QM1\ssl\key.kdb" -pw password -type  
cms -label ServerCert  
-file "Download_location\CERT.crt" -format ascii -trust enable
```

10. Add the new ReportingService stanza with the apikeyfile path to the queue manager's qm.ini file:

```
ReportingService:  
APIKeyFile=APIKey file location/apikeyfile.txt
```

11. Add the **API host** value to the qm.ini file.

The ReportingService stanza section now contains the path to the apikeyfile and the **API host (ServiceURL)** values:

```
ReportingService:  
APIKeyFile=APIKey file location/apikeyfile.txt  
ServiceURL=https://productinsights-api.ng.bluemix.net
```

Save and exit the qm.ini file.

12. Restart the queue manager for the changes to take effect.

You might be asked to grant permission for the queue manager process **amqzmur0** to access the network. The access is required to enable the queue manager to contact the metering service.

13. View the information about the queue manager *QM1* in your metering service instance.

When the reporting status is active, the startup and usage information for all integration servers on the specified integration node is reported to the metering service. The usage information is updated every 15 minutes.

14. Optional: Stop a queue manager from reporting to the metering service, by removing the ReportingService stanza from the queue manager's qm.ini file and restart the queue manager.
15. Optional: Check the diagnostic information in the queue manager's log file if the queue manager fails to report startup or usage information to the metering service.

AIX On AIX:

```
/var/mqm/qmgrs/QM1/errors/AMQERR0*.log
```

Linux On Linux:

```
/var/mqm/qmgrs/QM1/errors/AMQERR0*.log
```

Windows On Windows:

```
C:\ProgramData\IBM\MQ\errors\AMQERR0*.log
```

Results

You created a metering service instance and configured your queue manager to connect to the instance. You can see the information about your queue manager in the metering service instance dashboard.

Multi Connecting to the IBM Cloud Private metering service through an HTTP proxy

If your queue manager is running on a system that does not have direct access to your ICP cluster, you can use an HTTP proxy that your organization provides to connect to your metering service instance in IBM Cloud Private.

Before you begin

You have configured security, added the **API key** and service URL to the `qm.ini` file for your queue manager.

About this task

Use this task to configure your queue manager to connect to the [metering service](#) instance in IBM Cloud Private through an HTTP proxy that is provided by your organization.

Procedure

- Add a service proxy attribute to the IBM Cloud Private registration stanza of your `qm.ini` file. You can set the **ServiceProxy** attribute as follows:
 - A URL that includes the `http://` prefix and optionally the port. If you do not specify the port, `1080` is used.

```
ReportingService:
  ServiceProxy=http://myorgproxy.net:1080
```

Note: The **ServiceProxy** parameter must be set to a valid `http://` URL. Other proxy protocols, for example, HTTPS and SOCKS are not supported.

- Restart your queue manager before the changes take effect.

Multi Troubleshooting the connection to the metering service

Troubleshooting advice for errors that you might encounter when you are connecting your queue manager to a metering service instance.

Queue manager cannot register with or upload usage metrics to the configured metering service

Check that the queue manager has access to the network. The **APIKey** value in the API Key file is incorrect. Make sure that the IBM Global Security Kit (GSKit) component is installed.

Invalid qm.ini stanza

An invalid `qm.ini` stanza is found. Check the error log for more information.

Invalid HTTP service proxy parameter

The value for the **ServiceProxy** attribute for the queue manager `ReportingService` stanza is not configured correctly. The queue manager does not register with the service. The **ServiceProxy** parameter must be set to a valid `http://` URL. Other proxy protocols, for example, HTTPS and SOCKS are not supported.

Use these instructions to configure queue managers on IBM MQ for z/OS.

Before you begin

Before you configure IBM MQ for z/OS, read:

- [IBM MQ for z/OS concepts](#)
- [Planning your IBM MQ environment on z/OS](#)

About this task

After you have installed IBM MQ, you must carry out a number of tasks before you can make it available to users.

Procedure

- See the following subtopics for information on how to configure queue managers on IBM MQ for z/OS.

Related concepts

[z/OS](#) Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS

Related tasks

[“Creating queue managers on Multiplatforms” on page 7](#)

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

[Securing](#)

[“Configuring distributed queuing” on page 212](#)

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 16](#)

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](#) Administering IBM MQ for z/OS

[Planning](#)

Related reference

[z/OS](#) [Using the IBM MQ for z/OS utilities](#)

Preparing to customize queue managers on z/OS

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 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 System Modification Program Extended (SMP/E). For download links for the Program Directories, see [IBM MQ for z/OS Program Directory PDF files](#).

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:

- [“Setting up IBM MQ for z/OS” on page 957](#)
- [“Testing a queue manager on z/OS” on page 1021](#)
- [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 [Maintaining and migrating](#) 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 [Jakarta Messaging 3.0](#) or [Java Message Service 2.0](#) to connect to IBM MQ for z/OS.

See [IBM MQ for z/OS Program Directory PDF files](#) for information on installing IBM MQ for z/OS UNIX System Services Components.

IBM MQ for z/OS UNIX System Services Web Components

This feature is optional.

Select this feature if you want to use the IBM MQ Console, or the REST API.

You must install the IBM MQ for z/OS UNIX System Services Components feature, to install this feature.

IBM MQ for z/OS Managed File Transfer

This feature is optional, and should only be installed if you have entitlement for IBM MQ Advanced for z/OS, IBM MQ for z/OS Value Unit Edition (VUE), or IBM MQ for z/OS Managed File Transfer.

Select this feature if you want to use the Managed File Transfer capabilities of IBM MQ for z/OS.

You must install the IBM MQ for z/OS UNIX System Services Components feature, to install this feature.

Libraries that exist after installation

IBM MQ is supplied with a number of separate load libraries. [Table 71 on page 954](#) shows the libraries that might exist after you have installed IBM MQ.

<i>Table 71. IBM MQ libraries that exist after installation</i>	
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.
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.SCSQFCMD	Contains templates for jobs to create and run Managed File Transfer tasks.
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.

<i>Table 71. IBM MQ libraries that exist after installation (continued)</i>	
Name	Description
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.
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.

Table 71. IBM MQ libraries that exist after installation (continued)	
Name	Description
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).
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.SCSQTBK	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](#)

[“Using IBM MQ with IMS” on page 1059](#)

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 1067](#)

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 1069](#)

Use this topic if you want to use IMS Open Transaction Manager Access exits with IBM MQ for z/OS.

Related tasks

[“Setting up communications with other queue managers on z/OS” on page 1030](#)

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

[Administering IBM MQ for z/OS](#)

Related reference

[“Upgrading and applying service to Language Environment or z/OS Callable Services” on page 1067](#)

The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

z/OS

Setting up IBM MQ for z/OS

Use this topic as a step-by-step guide for customizing your IBM MQ for z/OS system.

The best way to configure a queue manager is to carry out the following steps in the order shown:

1. Configure the base queue manager.
2. Configure the channel initiator, which performs queue manager to queue manager communications, and remote client application communication.
3. If you want to encrypt or protect messages, configure Advanced Message Security for z/OS.
4. If you want to use IBM MQ to transfer files, configure Managed File Transfer for z/OS.
5. If you want to use the administrative or messaging REST API, or the IBM MQ Console to manage IBM MQ from a web browser, configure the mqweb server.

This topic leads you through the various stages of setting up IBM MQ after you have successfully installed it. The installation process is described in the Program Directory. For download links for the Program Directories, see [IBM MQ for z/OS Program Directory PDF files](#).

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 setting up 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.

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 [“Update SYS1.PARMLIB members” on page 971](#) 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](#).

Related concepts

[IBM MQ for z/OS concepts](#)

[“Using IBM MQ with IMS” on page 1059](#)

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 1067](#)

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 1069](#)

Use this topic if you want to use IMS Open Transaction Manager Access exits with IBM MQ for z/OS.

Related tasks

[“Setting up communications with other queue managers on z/OS” on page 1030](#)

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

[Administering IBM MQ for z/OS](#)

Related reference

[“Upgrading and applying service to Language Environment or z/OS Callable Services” on page 1067](#)

The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

Configuring the z/OS system for IBM MQ

Use these topics as a step by step guide for customizing your IBM MQ for z/OS system.

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(IEASYSp) 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 “APF authorize the IBM MQ load libraries” on page 958:

PROG=xx or APF=aa point to the Authorized Program Facility (APF) authorized library list (member PROGxx or IEFAPFaa)

For “Update the z/OS link list and LPA” on page 960:

LNK=kk points to the link list (member LNKLSTkk) LPA=mm points to the LPA list (member LPALSTmm)

For “Update the z/OS program properties table” on page 963:

SCH=xx points to the Program Properties Table (PPT) (member SCHEDxx)

For “Define the IBM MQ subsystem to z/OS” on page 964:

SSN=ss points to the defined subsystem list (member IEFSSNss)

APF authorize the IBM MQ load libraries

APF-authorize various libraries. Some load modules might already be authorized.

Notes:

- 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.
- Use of Library Look aside (LLA):

- Some IBM MQ usage can cause high Input/Output (IO) to load modules from libraries. This IO can be reduced by using the LLA facility of the operating system.
- This high IO can occur during:
 - Applications with a high MQCONN/MQDISC rate, for example in a WLM stored procedure.
 - Loading channel exits. If you have channels that start and stop frequently, and use channel exits.
- The member CSVLLAxx in SYS1.PARMLIB specifies the LLA setup. The inclusion of a library name in the LIBRARIES statement means that a program copy will always be taken from VLF(Virtual Lookaside Facility) and hence will not usually require I/O when heavily used.

Inclusion in the FREEZE statement means that there is no I/O to get the relevant DD statement concatenation directories (this can often be more I/O than the program load itself).

Use the operating system command " F LLA , REFRESH " after any changes to any of these libraries.

The following libraries must be APF-authorized:

- the IBM MQ load libraries thlqual.SCSQAUTH and thlqual.SCSQLINK
- the libraries for your national language feature thlqual.SCSQANLx and thlqual.SCSQSNLx, where x is the language letter for your national language:
 - C, for Chinese
 - E, for US English mixed case
 - F, for French
 - K, for Japanese
 - U, for US English uppercase
- the libraries for the distributed queuing feature thlqual.SCSQMVR1

All load modules in the LPA are automatically APF-authorized. Therefore they are all members of the link list if the SYS1.PARMLIB member IEASYSpp contains the following 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 [“Update the z/OS link list and LPA” on page 960](#)), 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 [APF-authorized library list](#) 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=h1q . SCSQ XXXX , VOLUME= YYYYYY for the changes to take effect: Where XXXX varies by the library name and where YYYYYY 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

[“Update the z/OS link list and LPA” on page 960](#)

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 queue managers on z/OS” on page 952](#)

Use this topic when customizing your queue managers with details of installable features, national language features, and information about testing, and setting up security.

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 should refresh the early code in each queue manager in the QSG to the IBM MQ 10.0.0 level before migrating any of the queue managers to IBM MQ 10.0.0.

Install the latest early code on each LPAR, and then refresh the queue managers one at a time at some point before migration. You do not have to migrate all of the queue managers at the same time.

- You might need to perform this task when migrating from a previous version. For further details, see the Program Directory. For download links for the Program Directories, see [IBM MQ for z/OS Program Directory PDF files](#).

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.

The early code comprises the following load modules:

- CSQ3INI and CSQ3EPX in the library thlqual.SCSQLINK
- CSQ3ECMX in the library thlqual.SCSQSNLx, where x is the language letter for your national language:
 - C, for Chinese
 - E, for US English mixed case
 - F, for French
 - K, for Japanese
 - U, for US English uppercase

IBM MQ includes a user modification that moves the contents of the thlqual.SCSQSNLx library into the thlqual.SCSQLINK and informs SMP/E. This user modification is called CSQ8UERL and is described in the *Program Directory for IBM MQ for z/OS*, for either Long Term Support or Continuous Delivery. For download links for the Program Directories, see [IBM MQ for z/OS Program Directory PDF files](#).

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 [“Define the IBM MQ subsystem to z/OS”](#) on page 964) 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=thlqual.SCSQLINK
SETPROG LPA,ADD,MODNAME=(CSQ3ECMX),DSNAME=thlqual.SCSQSNLx
```

- 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=thlqua1.SCSQLINK
```

- If you are using 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=thlqua1.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 existing queue managers using the following steps. Queue managers that you do not perform these steps on continue to use the version of early code that they are already using. It is not necessary to perform these steps for all queue managers on an LPAR, unless you are specifically trying to apply maintenance to all of them, or update them all to a more recent version or release of IBM MQ.

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
- SCSQANLx, where x is the language letter for your national language
- 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 MODIFY LLA REFRESH.

See [Modifying the contents of LNKLIST data sets](#) for more information, and [Using the dynamic LNKLIST facility safely and properly](#).

This is particularly recommended for SCSQAUTH so that you do not have to include it in several STEPLIBs. Only one language library, SCSQANLx should be placed in the LPA or link list. The link list libraries are specified in an LNKLISTkk 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.

Some modules are loaded at queue manager start up into ECSA. In ECSA constrained environments, you can place these modules into the LPA instead. See [“Placing IBM MQ global modules into the LPA” on page 962](#).

Related concepts

[“Update the z/OS program properties table” on page 963](#)

Some additional PPT entries are needed for the IBM MQ queue manager.

Placing IBM MQ global modules into the LPA

When an IBM MQ for z/OS queue manager starts up, it loads some of its load modules (global modules) into the extended common service area (ECSA). At queue manager shut down the ECSA is freed.

There are 19 global modules, which at IBM MQ 9.3, consumed approximately 1.2 MB of ECSA for each running queue manager.

Note: Although CSQ7GPLM is a global module, it should not be added to the LPA.

In environments that run multiple queue managers for each LPAR, and require a reduction in ECSA consumption due to ECSA or high private constraints, it is possible to place the global modules into the LPA. Placing the global modules of IBM MQ into the LPA is a manual process that requires care, so you should only carry out this procedure if there is a significant need to address ECSA or high private constraints.

If the queue manager cannot find a global module in its STEPLIB, and detects the module is in the LPA, it uses the LPA copy directly, instead of loading a copy of the module into ECSA. Alternatively if the queue managers code is normally loaded from the link list then any global modules in the LPA are loaded in preference to any global modules in the link list.

The z/OS common storage tracking function (see [Using the common storage tracking function](#)) tracks the storage under the each queue manager's MSTR address space for each queue manager and can be used to detect how much space is being use by the global modules.

By default, the global modules are in the SCSQAUTH load library. If the MSTR address space of a queue manager locates SCSQAUTH through the STEPLIB concatenation, the global modules from there are used in preference to any in the LPA and are loaded into ECSA.

The global modules are:

CSQ0GPLM, CSQ3AMGP, CSQ3SSGP, CSQ9PREP,
CSQ9SCNB, CSQGGPLM, CSQMCGLM, CSQMGPLM, CSQRGLM1,
CSQSLD1, CSQVGEPL, CSQVSRX, CSQWDL2, CSQWDL3,
CSQWVZSA, CSQWZDGO, CSQWVZPS, CSQWVGTM, CSQZTDDM

Important:

- The name of the global modules for IBM MQ remain constant across different IBM MQ versions. Therefore, if you load global modules into the LPA, they should be from a single IBM MQ version, and should only be used by queue managers running at the same IBM MQ version.
- If multiple versions of IBM MQ are run on the same LPAR then only one of those can have its global modules in the LPA at any given time.
- If maintenance is applied to an IBM MQ installation which has global modules loaded into the LPA, and that maintenance updates any of the global modules, you should perform the procedure described in the following text again.

Procedure

To put the global modules from a version of IBM MQ into the LPA, perform the following steps:

1. Create a copy of the thlqual.SCSQAUTH load library, and its contents, for example: thlqual.LOCAL.SCSQAUTH. Ensure that this load library is protected from unauthorized access using your external security manager (ESM).
2. APF authorize the thlqual.LOCAL.SCSQAUTH load library; see [“APF authorize the IBM MQ load libraries”](#) on page 958.
3. Create a new thlqual.GLOBAL.SCSQAUTH load library with the same attributes as thlqual.LOCAL.SCSQAUTH.

Note: This load library does not need to be APF authorized. Ensure that this load library is protected from unauthorized access using your ESM.

4. Copy the 19 global modules from `thlqual.LOCAL.SCSQAUTH` into `thlqual.GLOBAL.SCSQAUTH`.
5. Delete the 19 global modules from `thlqual.LOCAL.SCSQAUTH`.
6. Place the 19 global modules from `thlqual.GLOBAL.SCSQAUTH` into the LPA, by either:
 - a. Adding `thlqual.GLOBAL.SCSQAUTH` into an `LPALSTxx` member of `SYS1.PARMLIB`. You must then IPL the system with the `CLPA` option to ensure that the library contents are loaded into the PLPA.
 - b. Dynamically adding the modules to the LPA using the following command:

```
SETPROG
LPA,ADD,MODNAME=(CSQ0GPLM,CSQ3AMGP,CSQ3SSGP,CSQ9PREP,CSQ9SCNB,CSQGGPLM,
CSQMCGLM,CSQMGPLM,CSQRGLM1,CSQSLD1,CSQVGEPL,CSQVSRX,CSQWDL2,CSQWDL3,
CSQWVZSA,CSQWZDG0,CSQWVZPS,CSQWVGTM,CSQZTDDM),DSNAME= thlqual.GLOBAL.SCSQAUTH
```

Note: `LPALSTxx` is the preferred long term means of placing modules in LPA.

7. Validate that the modules are in the LPA by issuing the following command:

```
D PROG,LPA,MODNAME=CSQMCGLM
```

The output of the command should indicate the entry and load points of the module if it was successfully loaded into the LPA.

For each queue manager that needs to use the global modules from the LPA, then if you normally place:

1. `thlqual.SCSQAUTH` in the link list, just stop and start your queue manager. The global modules are loaded from the LPA, and the local modules from the link list.
2. `thlqual.SCSQAUTH` in the `MSTR JCL STEPLIB`, change the `JCL` so that the `STEPLIB` uses `thlqual.LOCAL.SCSQAUTH` instead of `thlqual.SCSQAUTH`. Stop and start the queue manager; the global modules are loaded from the LPA, and the local modules from the `STEPLIB`.

The `CHIN` and `AMSM JCL` can continue to use `thlqual.SCSQAUTH` as can any IBM MQ applications.

To revert the queue manager to loading the global modules into ECSA perform the following steps:

1. Stop the queue managers
2. Remove the global modules from the LPA, either at the next IPL by removing the `LPALSTxx` definitions or by using the following command:

```
SETPROG LPA,DELETE,MODNAME=(xxx) FORCE=YES
```

3. If `thlqual.LOCAL.SCSQAUTH` is in the `STEPLIB` of the queue manager replace it with `thlqual.SCSQAUTH`.
4. Restart the queue managers.

Related concepts

[“Update the z/OS link list and LPA” on page 960](#)

Update the LPA libraries with the new version of early code libraries. Other code can go in the link list or the LPA.

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 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, 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 Advanced Message Security, add the CSQ0DSRV PPT entry.

Issue the z/OS command **SET SCH=xx**, where xx is the suffix of the SCHEDxx member of PARMLIB, for these changes to take effect.

Related concepts

[“Define the IBM MQ subsystem to z/OS” on page 964](#)

Update the subsystem name table and decide on a convention for command prefix strings.

Configuring the queue manager and channel initiator

Use these topics as a step by step guide for configuring the queue manager and channel initiator.

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

[“Create procedures for the IBM MQ queue manager” on page 968](#)

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 [“Update SYS1.PARMLIB members” on page 971](#) 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 965 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 967 for information about system scope).

Figure 97 on page 965 shows several examples of IEFSSNss statements.

```
CSQ1,CSQ3INI,'CSQ3EPX,+mqs1cpf,S'  
CSQ2,CSQ3INI,'CSQ3EPX,+mqs2cpf,S'  
CSQ3,CSQ3INI,'CSQ3EPX,++,S'
```

Figure 97. 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 72 on page 965 gives a number of examples showing the associations of subsystem names and command prefix strings (CPFs), as defined by the statements in [Figure 97 on page 965](#).

Table 72. 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 74 on page 966](#).
- 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 74 on page 966](#)) 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 73 on page 966](#).

<i>Table 73. Example of CPF subset and superset rules</i>		
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 [“CPF in a sysplex environment” on page 967](#) for information about CPF registration).

[Table 74 on page 966](#) shows the characters that you can use when defining your CPF strings:

<i>Table 74. Valid character set for CPF strings</i>	
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.

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.

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 cataloged 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.

You need to include the library containing messages in your selected language. That is, thlqual.SCSQANLx, where x is the language letter for your national language:

- C, for Chinese
- E, for US English mixed case
- F, for French
- K, for Japanese
- U, for US English uppercase

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 limits of storage used by the queue manager, see [Storage configuration](#). 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.

Notes:

1. 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.
2. Sample started task procedures CSQ4MSTR and CSQ4MSRR have been updated to include, but leave commented out, the CSQMINI DD card that can be used to define a QMINI data set that contains transport security, that is, SSL or TLS properties.

You can use [“The QMINI data set” on page 974](#) to enable or disable TLS 1.3 support and/or be used to define a custom list of CipherSpecs to be used by channels.

Related concepts

[“Create procedures for the channel initiator” on page 969](#)

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.

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 modify the cataloged procedure 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 `xxxxx CHIN`, where `xxxxx` 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.

Consider adjusting the `MEMLIMIT` parameter using the information in [Storage configuration](#).

4. Authorize the procedures to run under your external security manager.
5. You need to include the library containing messages in your selected language. That is, `thlqual.SCSQANLx`, where `x` is the language letter for your national language:
 - C, for Chinese
 - E, for US English mixed case
 - F, for French
 - K, for Japanese
 - U, for US English uppercase

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 TLS, access to the system TLS 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`
- `/etc/resolv.conf` on the file system
- `//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

[“Define the IBM MQ subsystem to a z/OS WLM service class” on page 970](#)

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.

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 [Planning your IBM MQ environment on z/OS](#) for more information on setting performance options.

Related concepts

[“Set up the Db2 environment” on page 1007](#)

If you are using queue sharing groups you must create the required Db2 objects by customizing and running a number of sample jobs.

Implement your ESM security controls

Implement security controls for queue managers and the channel initiator.

- *Repeat this task for each IBM MQ queue manager.*
- *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 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 Transport Layer Security (TLS) or a sockets interface, ensure that the user ID under whose authority the channel initiator is running is configured to use z/OS UNIX System Services, as described in the [z/OS UNIX System Services Planning](#) documentation.
- If you are using TLS, 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.

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.
- Configuring z/OS data set encryption if required.

See the section, [confidentiality for data at rest on IBM MQ for z/OS with data set encryption](#), for more information.

For details about how to do this, see [Security installation tasks for z/OS](#).

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

[“Update SYS1.PARMLIB members” on page 971](#)

To ensure that your changes remain in effect after an IPL, you must update some members of SYS1.PARMLIB

[“Implement ESM security controls for the queue sharing group” on page 1010](#)

Implement security controls for all queue managers in a queue sharing group, to access Db2 and the coupling facility list structures.

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 [“Define the IBM MQ subsystem to z/OS” on page 964](#).
2. Change IEASYSpp so that the following members are used when an IPL is performed:
 - the PROGxx or IEAAPFaa members used in [“APF authorize the IBM MQ load libraries” on page 958](#)
 - the LNKLSTkk and LPALSTmm members used in [“Update the z/OS link list and LPA” on page 960](#)
 - the SCHEDxx member used in [“Update the z/OS program properties table” on page 963](#)
 - the IEFSSNss member used in [“Define the IBM MQ subsystem to z/OS” on page 964](#)

Related concepts

[“Customize the initialization input data sets” on page 971](#)

Make working copies of the sample initialization input data sets and tailor them to suit your system requirements.

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 DD names 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 DD names 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 th1qua1.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. The messages are divided into the following classes:

- System-related messages.
- Important long-lived messages.
- Short-lived messages.
- Miscellaneous messages.

Member CSQ4INPR uses multiple page sets for each of the major classes of message, and one page set for each other class. The following are the major classes of messages:

- Important long-lived messages.
- Short-lived messages.

Include the appropriate sample in the CSQINP1 concatenation of your queue manager started task procedure.

Notes:

1. IBM MQ supports up to 100 buffer pools 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.

- 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.

- 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 [Planning your page sets and buffer pools](#) 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 CSQ4IN*. CSQ4INY* members should 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 75. 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.
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 [Using the CSQUTIL utility for IBM MQ for z/OS](#).

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 76. Members of <i>thlqual.SCSQPROC</i>	
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 [Using the CSQUTIL utility for IBM MQ for z/OS](#).

Related concepts

[“Create the bootstrap and log data sets” on page 976](#)

Use the supplied program `CSQJU003` to prepare the bootstrap data sets (BSDSs) and log data sets.

The QMINI data set

You can use the `QMINI` data set to specify properties that are to be read and processed during queue manager initialization.

Characteristics of the QMINI data set

The `QMINI` data set is a sequential data set, or a member of a partitioned data set, with a maximum record length of 80 bytes (72 bytes for data and eight bytes for the line number).

The following example shows the properties for a sequential `QMINI` data set. Some properties are, of course, based on your environment.

```

Data Set Name . . . . : QM01.QMINI
General Data
Management class . . : STANDARD      Current Allocation
Storage class . . . : STANDARD      Allocated tracks . : 1
Volume serial . . . : P5P21E        Allocated extents . : 1
Device type . . . . : 3390
Data class . . . . . : **None**
Organization . . . . : PS            Current Utilization
Record format . . . : FB             Used tracks . . . . : 0
Record length . . . : 80             Used extents . . . . : 0
Block size . . . . . : 3120
1st extent tracks . : 1
Secondary tracks . . : 1
Data set name type . :
Data set encryption . : NO           Dates
SMS Compressible . . : NO           Creation date . . . : 2020/08/11
                                   Referenced date . . : ***None***
                                   Expiration date . . : ***None***

```

thlqual.SCSQPROC, includes:

- The sample contents for a QMINI data set in CSQ4QMIN.
- An example of specifying the QMINI data set using the //CSQMINI DD card, in the queue manager startup JCL, in the started task procedures CSQ4MSTR and CSQ4MSRR.

Notes:

- The code that parses the data set only parses the first 72 bytes of each record.
- Line numbers are ignored so it is not necessary to specify line numbers.
- If a line starts with an asterisk character (*), the line is treated as a comment.
- The contents of the QMINI data set are parsed during queue manager startup. If the contents are parsed successfully, message CSQM578I is issued in the queue manager job log. If any errors are encountered during parsing, error messages, for example [CSQM573E](#), are issued in the queue manager job log but the queue manager still starts.

Check for error messages, and resolve any issues in the contents of the QMINI data set.

If the queue manager is unable to parse the QMINI data set, you can start the channel initiator, but you cannot start any channels that are configured to use SSL or TLS as the security configuration settings are unknown.

- If you make any updates to the data set after you have started the queue manager, you must restart the queue manager to pick up the changes.

The TransportSecurity stanza

From IBM MQ for z/OS 9.2.0, the QMINI data set supports the TransportSecurity stanza. This stanza provides similar function to that provided by the SSL stanza in the qm.ini file on IBM MQ for Multiplatforms.

The TransportSecurity stanza supports the following properties:

AllowTLSV13

Whether a queue manager is able to use the TLS 1.3 CipherSpecs; valid values are: *TRUE/T/YES/Y* or *FALSE/F/NO/N*.

For migrated queue managers, TLS 1.3 is not enabled by default. You can enable TLS 1.3 by defining a QMINI data set with the TransportSecurity stanza and **AllowTLSV13=TRUE**.

For newly created queue managers TLS 1.3 is enabled by default.

AllowedCipherSpecs

A custom list of CipherSpecs that are enabled.

See [Providing a custom list of ordered and enabled CipherSpecs on IBM MQ for z/OS](#) for more information on this property.

Duplicate CipherSpec names in the list are ignored.

OutboundSNI

Whether the Server Name Indication (SNI) is set to the target IBM MQ channel name to the remote system when initiating a TLS connection, or to the hostname; valid values are: CHANNEL or HOSTNAME.

If the destination channel is configured with a certificate label on the channel object CERTLABL field, you must set CERTLABL to the channel value. If a connection with a setting of HOSTNAME is made to a channel with a CERTLABL setting, the connection fails and an AMQ9673 message is printed in the remote queue manager error logs.

The following example shows how the TransportSecurity stanza is specified:

```
TransportSecurity:
AllowedCipherSpecs=ECDHE_RSA_AES_128_GCM_SHA256,
                  ECDHE_RSA_AES_256_GCM_SHA384
AllowTLSV13=TRUE
```

The Channels stanza

— V10.0.0

From IBM MQ for z/OS 10.0.0, the QMINI data set supports the Channels stanza. This stanza provides the functionality of **IgnoreSeqNumberMismatch** within the same stanza in the `qm.ini` file on IBM MQ for Multiplatforms. The Channels stanza supports the following properties:

IgnoreSeqNumberMismatch

The Message Channel Agents (MCAs) at the two ends of a channel each keep count of the number of messages sent through the channel to maintain synchronization. Synchronization can be lost, for example if the channel definition at one end is deleted and then re-created. Under these circumstances, a **RESET CHANNEL** might be required to acknowledge that synchronization data has been lost and permit the channel to continue startup.

The **IgnoreSeqNumberMismatch** attribute must be set on the receiver queue manager. Effectively, this attribute performs a reset channel command on the receiver channel. This attribute controls how the queue manager handles a sequence number mismatch during channel startup using the following values:

NO

Channel sequence numbers are checked during channel resynchronization, if the two MCAs do not agree on the same sequence number, error message CSQX526 is reported and the channel fails to start.

YES

Channel sequence numbers are checked during channel resynchronization, but if the two MCAs do not agree on the same sequence number, warning message CSQX9703 is reported and the channel startup continues. This attribute value should not be needed under normal circumstances. When it is known that synchronization data has been lost, for example during disaster recovery, this option avoids needing to manually acknowledge each sequence number mismatch. Specifying this value has a similar effect to an administrator automatically issuing a **RESET CHANNEL** in response to each sequence number mismatch.

Create the bootstrap and log data sets

Use the supplied program CSQJU003 to prepare the bootstrap data sets (BSDSs) and log data sets.

Note:

- Repeat this task for each IBM MQ queue manager.
- If you are using z/OS data set encryption to protect the BSDS or active log data sets, you must configure this option before the data sets are allocated in this step.
- You do not need to perform this task when migrating from a previous version.
- If you are migrating a queue manager and adding z/OS data set encryption for active log data sets or BSDS, you need to convert the data sets.

- For more information about configuring z/OS data set encryption, and converting existing IBM MQ data sets to be encrypted, see [confidentiality for data at rest on IBM MQ for z/OS with data set encryption](#).

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 [“Create procedures for the IBM MQ queue manager”](#) on page 968, 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.

From 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.

Note: For queue managers created at IBM MQ 9.3.0 or later 8 byte log RBA is already enabled, so the following steps are not necessary.

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.

Note: If the queue manager is in a queue sharing group, all queue managers in the queue sharing group must have been started as follows before 8 byte log RBA can be enabled:

- If the queue manager is at IBM MQ 9.0.0 LTS it must have been started with **OPMODE(NEWFUNC,900)** or **OPMODE(NEWFUNC,800)**
- If the queue manager is at IBM MQ 9.0.n CD, or IBM MQ 9.1.0 LTS, or later, it needs to have been started at that level

Related concepts

[“Define your page sets”](#) on page 978

Define page sets for each queue manager using one of the supplied samples.

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.

If you are using z/OS data set encryption to protect the page sets, you must configure this option before the data sets are allocated in this step.

- You do not need to perform this task when migrating from a previous version.

If you are migrating a queue manager and adding z/OS data set encryption for page sets, you need to convert the page sets.

See the section, [Confidentiality for data at rest on IBM MQ for z/OS with data set encryption](#), for more information about configuring z/OS data set encryption and converting existing IBM MQ data sets to be encrypted.

Define separate page sets for each IBM MQ queue manager. thlqual.SCSQPROC(CSQ4PAGE) and thlqual.SCSQPROC(CSQ4PAGR) contain JCL and z/OS access method services (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 [Planning your page sets and buffer pools](#) for information about page sets and how to calculate suitable sizes.

The started task procedure CSQ4MSTR described in “[Create procedures for the IBM MQ queue manager](#)” on page 968 refers to the page sets, in a statement of the form:

```
//CSQP00nn DD DISP=OLD,DSN=xxxxxxxx
```

where *nn* is the page set number between 00 and 99, and *xxxxxxxx* 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.

See the [Optional Parameters](#) section of the z/OS DEFINE CLUSTER command for more information on REUSE.

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

[“Add the IBM MQ entries to the Db2 tables” on page 1010](#)

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.

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. You should create

your own system parameter module as some parameters, for example data set names, are usually site specific.

- Repeat this task for each IBM MQ queue manager, as required.
- You might need to perform this task when migrating from a previous version. For details, see [Migrating IBM MQ on z/OS](#).
- To enable 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 980](#). If you are configuring this queue manager for the first time, complete the whole of this task.

The system parameter module has four macros as follows:

Macro name	Purpose
CSQ6SYSP	Specifies the connection and tracing parameters, see “Using CSQ6SYSP” on page 980
CSQ6LOGP	Controls log initialization, see “Using CSQ6LOGP” on page 989
CSQ6ARVP	Controls archive initialization, see “Using CSQ6ARVP” on page 994
CSQ6USGP	Controls usage recording, see “Using CSQ6USGP” on page 1000

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 [Using MQSC to start and stop a queue manager on z/OS](#).

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

[“Tailor the channel initiator parameters” on page 1001](#)

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 77 on page 981](#). If you want to change any of these values, see the detailed descriptions of the parameters.

Table 77. Default values of CSQ6SYSP parameters

Parameter	Description	Default value	SET command
“ACCTIME” on page 982	The time, in minutes and seconds, between each gathering of accounting data.	-1	✓
“ACE LIM” on page 982	Size of ACE storage pool in 1 KB blocks.	0 (no limit)	✓
“CLCACHE” on page 983	Specifies the type of cluster cache to use.	STATIC	-
“CMDUSER” on page 984	The default user ID for command security checks.	CSQOPR	-
“EXCLMSG” on page 984	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 “Suppress information messages” on page 1006	()	✓
“EXITLIM” on page 984	Maximum time (in seconds) for which queue manager exits can run during each invocation.	30	-
“EXITTCB” on page 984	How many started server tasks to use to run queue manager exits.	8	-
“LOGLOAD” on page 984	Number of log records written by IBM MQ between the start of one checkpoint and the next.	500 000	✓
“MULCCAPT” on page 985	Determines the Measured Usage Pricing property which controls the algorithm for gathering data used by Measured Usage License Charging (MULC).	See parameter description	-
“OTMACON” on page 985	OTMA connection parameters.	See parameter description	-
“QINDXBLD” on page 986	Determines whether queue manager restart waits until all indexes are rebuilt, or completes before all indexes are rebuilt.	WAIT	-
“QMCCSID” on page 986	Coded character set identifier for the queue manager.	Zero	-
“QSGDATA” on page 986	Queue sharing group parameters.	See parameter description	-
“RESAUDIT” on page 987	RESLEVEL auditing parameter.	YES	-
“ROUTCDE” on page 987	Message routing code assigned to messages not solicited from a specific console.	1	-
“SERVICE” on page 987	Reserved for use by IBM.	0	✓

Table 77. Default values of CSQ6SYSP parameters (continued)

Parameter	Description	Default value	SET command
“SMFACCT” on page 987	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	-
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 Advanced Message Security for z/OS, set this parameter to YES.	NO	-
STATIME	The time, in minutes and seconds, 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	-

ACCTIME

Specifies the interval, in minutes and seconds, between consecutive gatherings of accounting data. Specify a number, either -1, or in the range 0 through 1440 minutes in the format 'mmm', or in the range 0 through 1440 minutes, and 0 - 59 seconds, in the format 'mmm.ss'.

Notes:

- When specifying an interval of seconds only, you must prefix the interval with a value of 0. The smallest possible interval is one second: '0.01'.
- If you specify a value of 0, accounting data is collected at the SMF global recording interval. See [Using System Management Facility](#) for more information.
- If you specify a value of -1, which is the default, accounting data is collected at the interval specified by the STATIME value.

For example:

'0.30' sets an interval of 30 seconds.

'5.30' sets an interval of 5 minutes and 30 seconds.

'30' sets an interval of 30 minutes.

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. The total amount of ECSA storage used is the sum of the QSRSPHBGF and QSRSPHBGV fields.

For more information about SMF 115 statistics records, see [Interpreting IBM MQ performance statistics](#).

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.

The cluster cache is an area of storage used to store information relating to the cluster.

If the cluster cache is static, it has a fixed size, which is allocated at queue manager start up. If the cache fills up then message **CSQM060E** is issued and the application request that required more space receives a **MQRC_CLUSTER_RESOURCE_ERROR**.

If you set **CLCACHE** to dynamic, the cluster cache can expand as required. However you first need to ensure that any installed cluster workload exits can function with a dynamic cache.

If an installed cluster workload exit can not function with a dynamic cache message **CSQM061E** is issued.

MQXCLWLN is provided for cluster workload exits to navigate the cluster cache in a way that works regardless of whether dynamic or static caches are used.

For new queue managers set **CLCACHE=DYNAMIC**, unless you are going to be using a cluster workload exit which does not support a dynamic cache.

For existing queue managers that already use a static cache and are in a cluster that is not having many new queues and queue managers added to it, it is reasonable to continue using **CLCACHE=STATIC**.

For existing queue managers that already use a static cache and are in a cluster that is going to have many new queues or queue managers added to it, start using **CLCACHE=DYNAMIC**.

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.

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 startup 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 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

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 1069](#).

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 code page listed as a native z/OS code page 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**.

Note: All queue managers in a queue sharing group should use the same **QMCCSID**.

QSGDATA

Queue sharing group data. This keyword takes five positional parameters:

QSGDATA=(Qsgname , Dsgname , Db2name , Db2serv , Db2b1ob)

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 *Routing codes* in [Message description](#) in one of the volumes of the *z/OS MVS System Messages* manuals.

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 data is gathered automatically in the range 1 through 4.

★ Start SMF accounting automatically for classes 1, 2 and 3.

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 5.

To gather class 2 or 3 statistics, class 1 must also be specified.

★ Start SMF statistics automatically for classes 1, 2 and 3.

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.

Security policy processing is configured for this queue manager, by setting **SPLCAP** to 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.

V 1000 E V 1000 The queue manager checks that the **AMSPROD** attribute is set to one of **ADVANCED** or **ADVANCEDVUE**, in which case it has been licensed for AMS. Otherwise it will default to using **ADVANCED**.

The queue manager also checks whether the necessary AMS configuration is in place. If it is not then the queue manager will not start.

If the queue manager is both licensed for AMS, and the necessary configuration is in place, the queue manager will start with message security capabilities enabled during queue manager initialization and the AMSM address space is started.

The default is NO.

STATIME

From IBM MQ for z/OS 9.3.0, specifies the time, in minutes and seconds, between consecutive gatherings of statistics data. If **ACCTIME** is not set, or is -1, also specifies the time, between consecutive gatherings of accounting data.

Specify a number in the range 0 through 1440 minutes in the format 'mmm', or in the range 0 through 1440 minutes, and 0 - 59 seconds, in the format 'mmm.ss'. The default is 30 minutes.

Notes:

- When specifying an interval of seconds only, you must prefix the interval with a value of 0. The smallest possible interval is one second: '0.01'.
- From IBM MQ for z/OS 9.3.0, if you specify a value of 0, statistics data is collected at the SMF data collection broadcast. If **ACCTIME** is not specified, or is -1, then accounting data is also collected at the SMF data collection broadcast. See [Using System Management Facility](#) for more information.
- If you specify a value of -1, which is the default, accounting data is collected at the interval specified by the **STATIME** value.

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 989](#)

Use this topic as a reference for how to specify logging options using CSQ6LOGP.

[“Using CSQ6ARVP” on page 994](#)

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.

<i>Table 78. Default values of CSQ6LOGP parameters</i>			
Parameter	Description	Default value	SET command
COMPLOG	Controls whether log compression is enabled.	NONE	X
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
ZHYWRITE	Specifies whether the zHyperWrite feature is enabled.	NO	X
ZHYLINK	Specifies whether the zHyperLink feature is enabled.	NO	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 128 through 40000.

The value specified is rounded up to a multiple of 4. Values between 40 and 128 will be accepted for compatibility reasons, and are treated as a value of 128.

The default is 4000 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.

ZHYWRITE

Specifies whether writes to the active logs are made with zHyperWrite being enabled.

For more information, see [Using zHyperWrite with IBM MQ active logs](#).

The value can be:

NO

zHyperWrite is not enabled.

YES

zHyperWrite is enabled.

ZHYLINK

Specifies whether writes to the active logs are made with zHyperLink being enabled.

For more information on enabling active logs with zHyperLink, see [Using zHyperLink with IBM MQ](#).

The value can be:

NO

zHyperLink is not enabled.

YES

zHyperLink is enabled.

Note: Enabling ZHYLINK also enables ZHYWRITE

Related reference

[“Using CSQ6SYSP” on page 980](#)

Use this topic as a reference for how to set system parameters using CSQ6SYSP.

[“Using CSQ6ARVP” on page 994](#)

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 79 on page 994](#). If you need to change any of these values, refer to the detailed descriptions of the parameters. For more information about planning your storage, see [Planning your storage and performance requirements on z/OS](#).

Table 79. 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
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

Table 79. Default values of CSQ6ARVP parameters (continued)

Parameter	Description	Default value	SET command
<u>UNIT</u>	Device type or unit name on which the first copy of archive log data sets is stored.	TAPE	X
<u>UNIT2</u>	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 [Planning your log archive storage](#).

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 *Routing codes* in [Message description](#) in one of the volumes of the *z/OS MVS System Messages* manuals.

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 two 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 [“Create the bootstrap and log data sets”](#) on page 976.
2. Determine the number of 4096 byte blocks in each archive log block:

$$d = \text{BLKSIZE} / 4096$$

where BLKSIZE is the rounded up value.

3. If ALCUNIT=BLK:

$$\text{PRIQTY} = \text{INT}(c / d) + 1$$

where INT means round down to an integer.

If ALCUNIT=TRK:

$$\text{PRIQTY} = \text{INT}(c / (d * \text{INT}(e/\text{BLKSIZE}))) + 1$$

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:

$$\text{PRIQTY} = \text{INT}(c / (d * \text{INT}(e/\text{BLKSIZE}) * f)) + 1$$

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 [“Create the bootstrap and log data sets”](#) on page 976 and [“Define your page sets”](#) on page 978.

Suggested settings:

Test system

1 680

Sufficient to hold the entire active log, that is:

$$10\ 080 / 6 = 1\ 680 \text{ blocks}$$

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, see [Planning your log archive storage](#).

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 [IBM z/OS Management Facility Programming Guide](#) for more information on ALCUNIT.

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 hhmsst.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 yyyddd.T hhmsst.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, see [Planning your log archive storage](#).

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 980](#)

Use this topic as a reference for how to set system parameters using CSQ6SYSP.

[“Using CSQ6LOGP” on page 989](#)

Use this topic as a reference for how to specify logging options using CSQ6LOGP.

 [Using CSQ6USGP](#)

Use this topic as a reference for how to set your system parameters using CSQ6USGP

Use CSQ6USGP to control product usage recording.

The default parameters for CSQ6USGP are shown in [Table 80 on page 1000](#). If you need to change any of these values, refer to the detailed descriptions of the parameters.



Attention: You cannot alter any of these parameters using the **SET SYSTEM** command.

Table 80. Default values of CSQ6USGP parameters		
Parameter	Description	Default value
QMGRPROD	Product against which queue manager usage is to be recorded	Blank
AMSPROD	Product against which Advanced Message Security (AMS) usage is to be recorded	 ADVANCED

QMGRPROD

Specifies the product against which queue manager usage is to be recorded.

Specify one of:

MQ

Queue manager usage is recorded as a stand-alone IBM MQ for z/OS product, with product ID 5656-MQA.

VUE

Queue manager usage is recorded as a stand-alone IBM MQ for z/OS Value Unit Edition (VUE) product, with product ID 5656-VUA.

ADVANCEDVUE

Queue manager usage is recorded as part of an IBM MQ Advanced for z/OS Value Unit Edition product, with product ID 5656-AVA.

AMSPROD

 Specifies the product against which Advanced Message Security usage is to be recorded, if used. If this parameter is not set, the default value of ADVANCED will be used.

Specify one of:

ADVANCED

AMS usage is recorded as part of an IBM MQ Advanced for z/OS product, with product ID 5656-ADA.

ADVANCEDVUE

AMS usage is recorded as part of an IBM MQ Advanced for z/OS Value Unit Edition product, with product ID 5656-AVA.

See [Reporting product information](#) for more information on product usage recording.

Related reference

[“Using CSQ6SYSP” on page 980](#)

Use this topic as a reference for how to set system parameters using CSQ6SYSP.

[“Using CSQ6LOGP” on page 989](#)

Use this topic as a reference for how to specify logging options using CSQ6LOGP.

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.

CHDISPS(20) is suitable for systems with more than 100 channels. There is unlikely to be any significant disadvantage in having CHDISPS(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 CHDISPS(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 CHDISPS.

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 ($\text{MIN}(\text{MAXCHL} / \text{CHDISPS}), 10$) 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 CHDISPS(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 $3 + \text{CHDISP} + \text{SSLTASKS}$ 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 CHDISPS 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 BPXPRMxx 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,0** command and you can use the **SETOMVS** command to dynamically change the MAXPROCUSER, or update the BPXPRMxx 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

[“Set up Batch, TSO, and RRS adapters” on page 1003](#)

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

Related reference

[Channel initiator statistics data records](#)

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.SCSQANLx
- thlqual.SCSQAUTH

where x is the language letter for your national language:

- C, for Chinese
- E, for US English mixed case
- F, for French
- K, for Japanese
- U, for US English uppercase

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

[“Set up the operations and control panels” on page 1004](#)

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.

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, where x is the language letter for your national language:
 - C, for Chinese
 - E, for US English mixed case
 - F, for French
 - K, for Japanese
 - U, for US English uppercase
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.

For the latter choice, use the z/OS [LIBDEF](#) command. See [Examples](#) for a link to various keywords you can use.

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 1005.

For more details, see the [z/OS: ISPF Dialog Developer's Guide and Reference](#).

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

[“Include the IBM MQ dump formatting member” on page 1006](#)

To be able to format IBM MQ dumps using the Interactive Problem Control System (IPCS), you must update some system libraries.

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 [z/OS MVS IPCS User's Guide](#) for more information.

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

[“Suppress information messages” on page 1006](#)

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.

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 `thlqual1.SCSQPROC(CSQ4MPFL)` shows suggested settings for `MPFLSTxx`. See [MPFLSTxx \(message processing facility list\)](#) for more information.

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, see [MVS Installation Exits](#).
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 980 for further information.

Related tasks

“[Testing a queue manager on z/OS](#)” on page 1021

When you have customized or migrated your queue manager, you can test it by running the installation verification programs and some of the sample applications shipped with IBM MQ for z/OS.

Configuring the queue sharing group

If you want to use shared queues for high availability, use these topics as a step by step guide for configuring the queue sharing group.

When you have completed the steps in this part of the process for setting up your IBM MQ for z/OS system, you should “[Tailor your system parameter module](#)” on page 978 to add queue sharing group data. You need to modify [CSQ6SYSP](#) to specify the QSGDATA parameter.

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.

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 need to perform the bind and grant steps 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 (UTS).

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 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 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 CSQ4XCTS to create the table spaces that contain the queue manager and channel initiator tables used for queue sharing groups.
4. Customize and execute sample JCL CSQ4XCTB to create the 15 Db2 tables and associated indexes. Do not change any of the row names or attributes.
5. Customize and execute sample JCL CSQ45BPL to bind the Db2 plans for the queue manager, utilities, and channel initiator.
6. Customize and execute sample JCL CSQ45GEX 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 shown in the following table.

User	Plans ()	Plans ()
Queue manager	CSQ5A 100, CSQ5C 100, CSQ5D 100, CSQ5K 100, CSQ5L 100, CSQ5M 100, CSQ5P 100, CSQ5R 100, CSQ5S 100, CSQ5T 100, CSQ5U 100, CSQ5W 100	CSQ5A 10X0, CSQ5C 10X0, CSQ5D 10X0, CSQ5K 10X0, CSQ5L 10X0, CSQ5M 10X0, CSQ5P 10X0, CSQ5R 10X0, CSQ5S 10X0, CSQ5T 10X0, CSQ5U 10X0, CSQ5W 10X0
SDEFS function of the CSQUTIL batch utility	CSQ52 100	CSQ52 10X0
CSQ5PQSG and CSQJUCNV batch utilities	CSQ5B 100	CSQ5B 10X0
CSQUZAP service utility	CSQ5Z 100	CSQ5Z 10X0

In the event of a failure during Db2 setup, the following jobs can be customized and executed:

- CSQ45DTB to drop the tables and indexes.
- CSQ4XDTS to drop the table spaces.
- CSQ4XDDDB to drop the database.
- CSQ4XDSDG 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 Administration](#) in *Db2 for z/OS 12.0.0* for more information about setting up Db2.

See [Planning on z/OS](#) for information about Db2 table sizes.

Related concepts

“Set up the coupling facility” on page 1008

If you are using queue sharing groups, define the coupling facility structures used by the queue managers in the queue sharing group (QSG) in the coupling facility Resource Management (CFRM) policy data set, using IXCMIAPU.

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 (QSG) in the coupling facility Resource Management (CFRM) policy data set, using IXCMIAPU.

See [Administrative data utility](#) for more information on 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
```

For information about planning CF structures and their sizes, see [Defining coupling facility resources](#).

Related concepts

[“Implement your ESM security controls” on page 970](#)

Implement security controls for queue managers and the channel initiator.

Set up the SMDS environment

If you want to use SMDS to offload messages on shared queues, set up the SMDS offload storage environment.

- *Perform this task for each queue manager and structure in the queue sharing group that you want to configure to offload data to SMDS.*
- *If you want to configure additional structures to offload data to SMDS later, this task can be performed again at that time.*
- *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.

Set up the SMDS environment

1. Estimate structure and data set space requirements. See [Shared message data set capacity considerations](#).
2. Allocate and preformat data sets. See [Creating a shared message data set](#).
3. When you define the CF structure to IBM MQ, ensure that you define the CFSTRUCT with CFLEVEL(5) and OFFLOAD(SMDS).

Related concepts

[“Set up the coupling facility” on page 1008](#)

If you are using queue sharing groups, define the coupling facility structures used by the queue managers in the queue sharing group (QSG) in the coupling facility Resource Management (CFRM) policy data set, using IXCMIAPU.

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.

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

[“Tailor your system parameter module” on page 978](#)

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. You should create your own system parameter module as some parameters, for example data set names, are usually site specific.

Implement ESM security controls for the queue sharing group

Implement security controls for all queue managers in a queue sharing group, to access Db2 and the coupling facility list structures.

- Repeat this task for each IBM MQ queue manager in a queue sharing group.
- You might need to perform this task when migrating from a previous version.

Ensure that the user IDs associated with the queue manager, channel initiator, and the utilities have authority to establish an RRSF connection to each Db2 subsystem with which you want to establish a connection. 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 RACF profile to which the user ID requires READ access is Db2ssid . RRSF in the DSNR resource class

The user IDs associated with each queue manager in a queue sharing group need to be granted the appropriate level of access to the coupling facility list structures. The RACF class is FACILITY.

The following user IDs require ALTER access:

- The queue manager ID to the IXLSTR . structure - name profile

- The user ID running CSQ5PQSG

Related concepts

[“Implement your ESM security controls” on page 970](#)

Implement security controls for queue managers and the channel initiator.

Configuring Advanced Message Security for z/OS

Use these topics as a step by step guide for configuring Advanced Message Security (AMS).

Before you begin

Before you start to configure AMS, ensure that the following queue manager configuration steps have been performed:

1. Add the CSQ0DRTM module to the LPA, as described in [“Update the z/OS link list and LPA” on page 960](#).
2. Add an entry for CSQ0DSRV to the z/OS program properties table (PPT), as described in [“Update the z/OS program properties table” on page 963](#).
3. Include the CSQ4INSM member in the CSQINP2 concatenation of queue manager started task procedure, as described in [“Customize the initialization input data sets” on page 971](#).
4. Enable AMS using the AMSPROD attribute. See [product usage recording with IBM MQ for z/OS products](#) for more details.

What to do next

Configure policies for queues protected by AMS. Security policies are described in [Administering Advanced Message Security security policies](#).

There are examples of AMS configurations in [Example configurations on z/OS](#).

Creating procedures for Advanced Message Security

Each IBM MQ subsystem that is to be configured to use Advanced Message Security (AMS) requires a cataloged procedure to start the AMS address space. You can create your own or use the IBM-supplied procedure library.

Procedure

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.
5. Repeat steps 1 to 4 for each IBM MQ queue manager.

What to do next

[“Setting up the Advanced Message Security started task user ID” on page 1012](#)

Setting up the Advanced Message Security started task user ID

The Advanced Message Security (AMS) task requires a user ID that allows it to be known as a z/OS UNIX System Services (z/OS UNIX) process.

About this task

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 z/OS UNIX System Services users. For more information on defining z/OS UNIX System Services UIDs and GIDs, see [z/OS: Security Server RACF Security Administrator's Guide](#).

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 Advanced Message Security task according to your installation's security policy for deploying and running privileged z/OS UNIX System Services processes.

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 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 Advanced Message Security task program (*thlqual.SCSQAUTH(CSQ0DSRV)*) must be located in RACF program-controlled libraries.

Note: Choose the user ID for this task carefully because the 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 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](#).

Procedure

1. Define the Advanced Message Security started task user to RACF. The examples in this section use the user ID WMQAMSM.

```
ADDUSER WMQAMSM NAME('AMS user') OMVS (UID(0)) DFLTGRP(group)
```

Select a default 'group' as appropriate to your installation standards.

Note: If you do not want to grant z/OS UNIX superuser authority (UID(0)), then you must permit the 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 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 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 Advanced Message Security tasks, specifying the user ID you selected or created in step 1:

```
RDEFINE STARTED qmgrAMSM.* STDATA(USER(WMQAMSM))
```

where *qmgr* is the prefix of the started task name. For example, the started task may be named CSQ1AMSM. In this case, you would substitute *qmgrAMSM.** with CSQ1AMSM.*.

The AMS started tasks must be named *qmgrAMSM*.

4. Use the **SETROPTS** RACF command to refresh the in-storage RACLISTed STARTED class profiles:

```
SETROPTS RACLIST(STARTED) REFRESH
```

5. The Advanced Message Security task temporarily assumes the identity of the host user ID of the 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.

If the RACF SURROGAT class is active, defining a single generic profile allows the Advanced Message Security task to assume the identity of any user. 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 generic profile in the SURROGAT class:

```
RDEFINE SURROGAT BPX.SRV.* UACC(NONE)
```

- e) Permit the Advanced Message Security user ID to the generic SURROGAT class profile:

```
PERMIT BPX.SRV.* CLASS(SURROGAT) ID(WMQAMSM) ACCESS(READ)
```

Note: You can define more specific profiles if you want to restrict specific users to be processed by the Advanced Message Security task, as described in [z/OS UNIX System Services Planning](#).

For example, a profile called BPX.SRV.MQUSER1 controls whether the AMS task can assume the identity of the user ID MQUSER1.

- f) Permit the 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)
```

- g) Use the **SETROPTS** RACF command to refresh the in-storage RACLISTed started class profiles:

```
SETROPTS RACLIST(SURROGAT) REFRESH  
SETROPTS RACLIST(FACILITY) REFRESH
```

6. The 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 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.* CLASS(FACILITY) ID(WMQMSM) ACC(READ)
```

Alternatively, you can grant READ access to the data service task user's keyring in the RDATA LIB class as follows:

```
PERMIT WMQASMD.DRQ.AMS.KEYRING.LST CLASS(RDATA LIB) ID(WMQMSM) ACC(READ)
```

7. Configure resource security:

- a) The Advanced Message Security started task user requires authority to connect to the queue manager as a batch application.

If your queue manager has connection security enabled, grant the AMS task authority to connect to the queue manager with this command:

```
PERMIT hlq.BATCH CLASS(MQCONN) ID(WMQMSM) ACC(READ)
```

where *hlq* can be either the queue manager name queue sharing group name.

For further information, see [Connection security profiles for batch connections](#).

- b) The Advanced Message Security started task user requires authority to browse the SYSTEM.PROTECTION.POLICY.QUEUE.

If queue security is active on the queue manager, grant the AMS user authority to access the queue with these commands:

```
RDEFINE MQQUEUE hlq.SYSTEM.PROTECTION.POLICY.QUEUE UACC(NONE)
PERMIT hlq.SYSTEM.PROTECTION.POLICY.QUEUE CLASS(MQQUEUE) ID(WMQMSM) ACCESS(READ)
```

where *hlq* can be either the queue manager name queue sharing group name.

If the queue manager is using mixed case profiles, define the profile in the MXQUEUE class instead.

To manage AMS security policies using the CSQOUTIL utility, administrators need access to put messages to the SYSTEM.PROTECTION.POLICY.QUEUE. This is performed by granting UPDATE access to the profile protecting the queue.

For further information, see [Profiles for queue security](#).

What to do next

[“Granting RACDCERT permissions to the security administrator for Advanced Message Security” on page 1015](#)

Granting RACDCERT permissions to the security administrator for Advanced Message Security

Your Advanced Message Security security administrator requires authority to use the RACDCERT command to create and manage digital certificates.

Procedure

- 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 Advanced Message Security security administrator.

What to do next

[“Grant users resource permissions for Advanced Message Security” on page 1015](#)

Grant users resource permissions for Advanced Message Security

Advanced Message Security users require relevant resource permissions.

About this task

Advanced Message Security users, that is users that are putting or getting Advanced Message Security protected messages, require:

- An OMVS segment associated with their user id
- Permissions for IRR.DIGTCERT.LISTRING or RDATA LIB
- Permissions for ICSF class CSFSERV and CSFKEYS profiles
- Permission to put to the SYSTEM.PROTECTION.ERROR.QUEUE

The 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 Advanced Message Security during the processing of IBM MQ messages to queues that are protected by 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 Advanced Message Security, you might 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.

Procedure

- Grant READ permission to the IRR.DIGTCERT.LISTRING profile in the FACILITY class:

- To grant READ permission to the IRR.DIGTCERT.LISTRING profile in the FACILITY class to all users, issue this command:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(READ)
```

- To grant READ permission to the IRR.DIGTCERT.LISTRING profile in the FACILITY class on a per user basis, issue this command:

```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(userid) ACCESS(READ)
```

where `userid` is the name of the Advanced Message Security user.

- Alternatively, use the `RDATA LIB` class to grant access to specific key rings. The `RDATA LIB` permissions take precedence over `IRR.DIGTCERT.LISTRING` permissions. For example:

```
PERMIT user.DRQ.AMS.KEYRING.LST CLASS(RDATA LIB) ID(user) ACC(READ)
```

- If you are using ICSF-managed certificates and private keys, Advanced Message Security users require access to certain class `CSFSERV` and `CSFKEYS` profiles. This access is detailed in the following table:

Table 81. Required user access to class <code>CSFSERV</code> and <code>CSFKEYS</code> profiles		
Class	Profile	Permission
<code>CSFSERV</code>	<code>CSFDSG</code>	READ
<code>CSFSERV</code>	<code>CSFPKE</code>	READ
<code>CSFSERV</code>	<code>CSFPKD</code>	READ
<code>CSFSERV</code>	<code>CSFDSV</code>	READ
<code>CSFKEYS</code>	ICSF PKDS Label	READ

- Applications that perform operations on queues with AMS policies defined need access to put messages to `SYSTEM.PROTECTION.ERROR.QUEUE`. Grant put access to the queue with these commands:

```
RDEFINE MQQUEUE hlq.SYSTEM.PROTECTION.ERROR.QUEUE UACC(NONE)
PERMIT hlq.SYSTEM.PROTECTION.ERROR.QUEUE CLASS(MQQUEUE) ID(userID) ACCESS(UPDATE)
```

where `hlq` can be either the queue manager name queue sharing group name, and `userID` is the application user ID.

What to do next

[“Creating key rings for Advanced Message Security” on page 1016](#)

Creating key rings for Advanced Message Security

Certificates used by Advanced Message Security (AMS) for signing and encryption are stored in z/OS SAF key rings. You need to create these key rings and certificates before you can use AMS.

About this task

Advanced Message Security accesses certificates in the following key rings:

- A single key ring owned by the AMS address space user.
- Key rings owned by the individual users that send or receive messages on queues with AMS policies defined.

These key rings must all be named `drq.ams.keyring`.

There is more information on key rings and certificates used by AMS, and an example scenario, in [Using certificates on z/OS](#).

Follow these steps to create the key rings required by AMS, and connect certificates to the key rings. You must create the key ring owned by the AMS address space user before starting AMS. You can create the keys rings owned by the users that send or receive messages at any time.

Procedure

- Issue the following command to create a key ring owned by the AMS address space user:

```
RACDCERT ID(amsUser) ADDRING(drq.ams.keyring)
```

where *amsUser* is the user ID of the AMS address space.

2. Create a key ring for each user that sends or receives messages protected by AMS by issuing the command in step 1 for each user ID.
3. Connect the certificate authority (CA) certificate for the issuer of the user certificates to the key ring owned by the AMS address space user ID. Issue the following command:

```
RACDCERT ID(amsUser) CONNECT(CERTAUTH LABEL('caLabel') RING(drq.ams.keyring))
```

where *amsUser* is the user ID of the AMS address space, and *caLabel* is the label of the CA certificate.

If you are using RACF as your CA, and need to create a certificate authority certificate, follow the example in [Defining a local Certificate Authority certificate](#).

4. If you are using privacy or confidentiality security policies to encrypt messages on queues protected by AMS, connect the certificates of message recipients to the key ring owned by the AMS address space user ID. Issue the following command:

```
RACDCERT ID(amsUser) CONNECT(ID(userId) LABEL('certLabel')  
RING(drq.ams.keyring) USAGE(SITE))
```

where *amsUser* is the user ID of the AMS address space, *userId* is the message recipient, and *certLabel* is the label of the user's certificate.

The USAGE(SITE) attribute prevents the private key from being accessible in the key ring.

If you are creating your own certificates with RACF, follow the example in [Creating a digital certificate with a private key to create the certificate](#).

5. Connect the certificates of each user that sends or receives messages protected by AMS to a key ring owned by the user. The certificate must be connected as the default certificate in the key ring. Issue the following command:

```
RACDCERT ID(userId) CONNECT(ID(userId) LABEL('certLabel')  
RING(drq.ams.keyring) DEFAULT USAGE(PERSONAL))
```

where *userId* is the user that is sending or receiving messages, and *certLabel* is the label of the user's certificate.

Notes:

- a. Steps “2” on page 1017 and “5” on page 1017 are not required if the application opens a queue only for output, and sends messages to queues protected by an AMS confidentiality policy.
- b. Steps “2” on page 1017 and “5” on page 1017 are not required if the application opens a queue only for input/browse, and receives messages from queues protected by an AMS integrity policy.

What to do next

[“Enabling Advanced Message Security” on page 1017](#)

Enabling Advanced Message Security

Security policy capability for a queue manager is controlled by the SPLCAP parameter in the system parameter module.

About this task

Follow these steps to enable Advanced Message Security (AMS) for a single queue manager.

This task requires you to make a change to the system parameter module. See [“Tailor your system parameter module” on page 978](#) for more information on creating and customizing the system parameter module.

Procedure

1. Set **SPLCAP** to YES in CSQ6SYSP. See [“Using CSQ6SYSP” on page 980](#) for more information on the CSQ6SYSP macro.
2.  Set the **AMSPROD** to either ADVANCED, or ADVANCEDVUE depending on your licence entitlement. See [using CSQ6USGP](#) for more information on the CSQ6USGP macro.
3. Recompile the system parameter module.
4. Restart the queue manager with the updated system parameter module. The AMS address space is started automatically when the queue manager starts.

Configuring the mqweb server

Use these topics as a step by step guide for configuring the mqweb server.

Related tasks

[“Configuring the IBM MQ Console and REST API” on page 906](#)

The mqweb server that hosts the IBM MQ Console and REST API is provided with a default configuration. In order to use either of these components a number of configuration tasks need to be completed, such as configuring security to allow users to log in. This topic describes all the configuration options that are available.

crtmqweb (creating the mqweb server)

If you installed the IBM MQ for z/OS UNIX System Services Web Components, and want to use the IBM MQ Console, or the REST API, use **crtmqweb** to create and customize the mqweb server.

Before you begin

Before you run the **crtmqweb** script to create the mqweb server, set the **JAVA_HOME** environment variable to reference a 64 bit version of Java on your system.

Note:  From IBM MQ 10.0, the Java runtime environment (JRE) on z/OS has support for IBM Semeru Runtime 21 and IBM Semeru Runtime 25.

The IBM MQ Console and the administrative REST API require the SYSTEM.REST.REPLY.QUEUE queue to be created. Create this queue by using the **CSQ4INSG** sample in [“Customize the initialization input data sets” on page 971](#).



Attention: When starting the mqweb server, if you encounter error message CWWKG0014E, as displayed in the following output:

```
Launching mqweb (MQM MVS/ESA V9 R2.0/wlp...) (en_US)
YAUDIT    CWWKE0001I: The server mqweb has been
launched.
          YWARNING  " CWWKF0009W: The server has not been configured to install any
features.
          YAUDIT    " CWWKF0011I: The mqweb server is ready to run a smarter planet.
The mqweb server started in 6.348 seconds.
          YERROR    " CWWKG0014E: The configuration parser detected an XML syntax
error while parsing the root of the configuration and the referenced configuration
documents.
                                Error: An invalid XML character (Unicode: 0x4c) was found
in the prolog of the document.
                                File: file:<your filepath>/servers/mqweb/server.xml Line:
1 Column: 1
```

you should check the z/OS setting of AUTOCVT (automatically convert files from one code set to another) and adjust the value as required by doing one of the following.

In a USS terminal:

Issue the command: `echo $_BPXK_AUTOCVT` to display the value of this environment variable. If the environment variable is not defined, no value is displayed.

To set the environment variable, see [_BPXK environment variables](#).

System wide:

Example 6 at [Displaying status of z/OS UNIX System Services \(OMVS\)](#) shows how to display the value of the system wide AUTOCVT statement in BPXPRMxx.

To set the environment variable system wide, use the [AUTOCVT](#) statement in BPXPRMxx.

If environment variable `_BPXX_AUTOCVT` is set in a USS terminal, it overrides the system wide setting of statement AUTOCVT in BPXPRMxx.

About this task

- Complete this task once for each z/OS system where you want to run the IBM MQ Console or REST API.
- To use the administrative REST API, you need a mqweb server for each version of IBM MQ that is running. For example, if you are running IBM MQ 9.4.0, IBM MQ 9.3.5 and IBM MQ 9.3.0, you need three different mqweb servers.
- You might need to refresh or modify the server configuration when you migrate from a previous version.

The IBM MQ Console and REST API require the creation of a single WebSphere Liberty server, called mqweb.

The server configuration and log files are all stored under the Liberty user directory.

The mqweb server needs to be configured with a product ID (PID) that it runs under. The PID is set when the mqweb server is created. Use the same PID that is used to run the local queue managers that the mqweb server connects to.

Note: If local queue managers run under multiple different PIDs, choose one of them to run the mqweb server under.

For more information about PIDs and how they are used on z/OS, see [Product usage recording with IBM MQ for z/OS products](#).

It is possible to change the PID that the mqweb server runs under, after it is created, by using the [setmqweb pid](#) command.

Complete the following steps to create the mqweb server:

Procedure

1. Decide which PID the mqweb server runs under.
2. Choose a suitable location for the Liberty user directory.

The user ID that the mqweb server runs under, needs read and write access to this user directory and its contents. As this user directory contains log files, in addition to the server configuration, create this directory in a separate file system.

Note: There is a significant amount of disk I/O when the mqweb server starts. To reduce the time taken to start the mqweb server, ensure that both the IBM MQ installation z/OS UNIX file system and the Liberty user directory file system are either sysplex-aware, or locally mounted on the system where the mqweb server is running.

3. In z/OS UNIX System Services, change your current working directory to `PathPrefix/web/bin` by issuing the following command:

```
cd PathPrefix/web/bin
```

where *PathPrefix* is the IBM MQ for z/OS UNIX System Services Components installation path.

4. Create the Liberty user directory that contains the template mqweb server definition, by running the **crtmqweb** script.

The format of the **crtmqweb** command is:

```
crtmqweb user_directory -p pid_value
```

where:

user_directory

Is the Liberty user directory decided in step “2” on page 1019. This parameter is optional. If this parameter is not specified, then a default Liberty user directory of /var/mqm/web/installation1 is used.

pid_value

Indicates the PID that the mqweb server runs under. This PID is the one that you chose in step “1” on page 1019. *pid_value* is one of the following values:

MQ

The mqweb server runs under PID IBM MQ for z/OS (5656-MQA).

VUE

The mqweb server runs under PID IBM MQ for z/OS Value Unit Edition (VUE) (5656-VUA).

ADVANCEDVUE

The mqweb server runs under PID IBM MQ Advanced for z/OS VUE (5656-AVA),

For example, to create the mqweb server with a Liberty user directory of /usr/mqweb and a PID of IBM MQ Advanced for z/OS VUE (5656-AVA), run the following command:

```
./crtmqweb /usr/mqweb -p ADVANCEDVUE
```

5. Change the ownership of the directories and files in the Liberty user directory, so that they belong to the user ID and group that the mqweb server runs under, by using the command:

```
chown -R userid:group path
```

To give the group write access to the path, issue the command:

```
chmod -R 770 path
```

What to do next

[“Creating a procedure for the mqweb server” on page 1020](#)

Related tasks

[“Configuring the IBM MQ Console and REST API” on page 906](#)

The mqweb server that hosts the IBM MQ Console and REST API is provided with a default configuration. In order to use either of these components a number of configuration tasks need to be completed, such as configuring security to allow users to log in. This topic describes all the configuration options that are available.

Creating a procedure for the mqweb server

If you installed the IBM MQ for z/OS UNIX System Services Web Components, and want to use the IBM MQ Console, or the REST API, you need to create a cataloged procedure to start the mqweb server. The mqweb server is a Liberty server that hosts the IBM MQ Console and the REST API.

- You need to perform this task once for each z/OS system where you want to run the IBM MQ Console or REST API.
- You need a mqweb server for each version of IBM MQ that is running. For example, a started task called MQWB0910 for queue managers at IBM MQ for z/OS 9.1.0 and a started task called MQWB0905 for queue managers at IBM MQ for z/OS 9.0.5.

If you have only one queue manager on the z/OS system, you can run a single Liberty server started task, and change the libraries it uses when you migrate your queue manager.

- You might need to modify the cataloged procedure when migrating from a previous version.

Carry out the following procedure to create a cataloged procedure:

1. Copy the sample started task procedure th1qua1.SCSQPROC (CSQ4WEBS) to your procedure library.
Name the procedure according to the standards of your enterprise.

For example MQWB0910, indicating that this is the cataloged procedure for the IBM MQ for z/OS 9.1.0 mqweb server.

2. Tailor the procedure to your requirements using the instructions in the sample procedure CSQ4WEBS.

Note that the Liberty user directory is the directory specified when the **crtmqweb** script was run to create the mqweb server definition.

See [“crtmqweb \(creating the mqweb server\)” on page 1018](#) for details.

Note: Ensure that you specify **Caps off** when you edit the member, as the file has lowercase data.

3. Authorize the procedure to run under your external security manager.
4. Use IBM Workload Manager (WLM) to classify this address space.

The mqweb server is an IBM MQ application, and users interact with this application. The application does not need to be high importance in WLM, and a service class of **STCUSER** might be suitable.

What to do next

Follow the steps in [“Basic configuration for the mqweb server” on page 907](#) to finish configuring the mqweb server.

Related tasks

[“Configuring the IBM MQ Console and REST API” on page 906](#)

The mqweb server that hosts the IBM MQ Console and REST API is provided with a default configuration. In order to use either of these components a number of configuration tasks need to be completed, such as configuring security to allow users to log in. This topic describes all the configuration options that are available.

Testing a queue manager on z/OS

When you have customized or migrated your queue manager, you can test it by running the installation verification programs and some of the sample applications shipped with IBM MQ for z/OS.

About this task

After you have installed and customized IBM MQ for z/OS, you can use the supplied installation verification program, CSQ4IVP1, to confirm that IBM MQ for z/OS is operational.

The basic installation verification program CSQ4IVP1 tests non-shared queues and verifies the base IBM MQ without using the C, COBOL, or CICS samples.

After running the basic installation verification, you can test for shared queues by using CSQ4IVP1 with different queues, and also test that Db2 and the coupling facility are set up correctly. To confirm that distributed queuing is operational, you can use the supplied installation verification program, CSQ4IVPX,

CSQ4IVP1 is supplied as a load module, and provides a set of procedural sample applications as source modules that demonstrate typical uses of the Message Queue Interface (MQI). You can use these source modules to test different programming language environments. You can compile and link-edit whichever of the other samples are appropriate to your installation by using the supplied sample JCL supplied.

Procedure

- For information on how to test your queue manager on z/OS, see the following subtopics:
 - [“Running the basic installation verification program” on page 1022](#)
 - [“Testing for queue sharing groups” on page 1025](#)
 - [“Testing for distributed queuing” on page 1026](#)
 - [“Testing for C, C++, COBOL, PL/I, and CICS programs with IBM MQ for z/OS” on page 1029](#)

Related concepts

[IBM MQ for z/OS concepts](#)

Related tasks

[Planning your IBM MQ environment on z/OS](#)

[“Configuring queue managers on z/OS” on page 952](#)

Use these instructions to configure queue managers on IBM MQ for 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 98 on page 1023](#). 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 98. 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 99 on page 1025](#)

 For an explanation of the completion and reason codes, see the [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 QUALIFIER : 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 PERFORMANCE 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 99. Sample report from CSQ4IVP1

z/OS 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 “[Set up the coupling facility](#)” on [page 1008](#). 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 100 on page 1026](#). 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 100. 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 1023](#) 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 101 on page 1028](#). 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 101 on page 1028](#).

```

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.SYNCO
PERMIT CSQ1.SYSTEM.CHANNEL.SYNCO 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 101. 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 102 on page 1029](#), 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 TCP/IP
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 102. Example output from CSQ4IVPX

Testing for C, C++, COBOL, PL/I, and CICS programs with IBM MQ for z/OS

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 [Using the sample procedural programs for z/OS](#).

Setting up communications with other queue managers on z/OS

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

About this task

To define your distributed-queuing requirements, you need to define the following items:

- The channel initiator procedures and data sets
- The channel definitions
- The queues and other objects
- Access security

If you are using queue sharing groups, see [Distributed queuing and queue sharing groups](#).

For additional points to consider when you are preparing to set up distributed queuing with IBM MQ for z/OS, see [“Considerations for using distributed queuing on z/OS” on page 1030](#).

Procedure

To enable distributed queuing, complete the following steps:

- Customize the distributed queuing facility and define the IBM MQ objects required as described in [Defining system objects](#) and [“Preparing to customize queue managers on z/OS” on page 952](#).
- 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 1049](#).

Related concepts

[“Setting up IBM MQ for z/OS” on page 957](#)

Use this topic as a step-by-step guide for customizing your IBM MQ for z/OS system.

Related tasks

[“Configuring distributed queuing” on page 212](#)

This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Considerations for using distributed queuing on z/OS

Points to consider when you are preparing to use distributed queuing on z/OS.

If you are using queue sharing groups, see [Distributed queuing and queue sharing groups](#).

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 [CSQ9023E](#). 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. For more information, see [The dead-letter queue handler utility \(CSQUDLQH\)](#).

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. For more information, see [Security considerations for the channel initiator on z/OS](#), [Profiles for queue security](#), and [“Implement your ESM security controls” on page 970](#).

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 more 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 more information, see [Using the z/OS Automatic Restart Manager \(ARM\)](#).

Related concepts

[“Setting up IBM MQ for z/OS” on page 957](#)

Use this topic as a step-by-step guide for customizing your IBM MQ for z/OS system.

Related tasks

[“Configuring distributed queuing” on page 212](#)

This section provides more detailed information about intercommunication between IBM MQ installations, including queue definition, channel definition, triggering, and sync point procedures

Defining IBM MQ objects on z/OS

On z/OS, use one of the IBM MQ command input methods to define IBM MQ objects.

For more information about defining objects, see [“Monitoring and controlling channels on z/OS” on page 1033](#).

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.

 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 235](#).

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:

<i>Table 82. Channel tasks</i>	
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 1035](#)

You can use the MQSC commands, the PCF commands, or the operations and control panels to manage DQM.

[“Setting up IBM MQ for z/OS” on page 957](#)

Use this topic as a step-by-step guide for customizing your IBM MQ for z/OS system.

[“Setting up communication for z/OS” on page 1049](#)

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 1054](#)

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 1058](#)

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.

Related tasks

[“Configuring distributed queuing” on page 212](#)

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 z/OS” on page 1030](#)

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

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 MQSC commands, see [Administering IBM MQ using MQSC commands](#). For information about PCF commands, see [Automating administration using Programmable Command Formats commands](#).

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 103 on page 1035](#) 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, 2026. All rights reserved.

Command ==> -----
F1=Help F2=Split F3=Exit F4=Prompt F9=SwapNext F10=Messages
F12=Cancel
```

Figure 103. 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 104 on page 1036](#) 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 104. Listing channels

Defining a channel on z/OS

On z/OS, you can define a channel by using MQSC commands or using the operations and control panels.

Procedure

- To define a channel using the MQSC commands, use the **DEFINE CHANNEL** command.
- To use the operations and control panels, starting from the initial panel, complete the following fields and press Enter:

Table 83. Operations and control panels: initial panel fields	
Field	Value to enter in field
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 that you want for the channel you are defining. They are initialized with the default attribute values. Change any that you want to 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.

What to do next

After you have defined your channel you must secure your channel. For more information, see [“Securing a channel”](#) on page 1038.

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 105 on page 1039](#). 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 105. Command Entry

The output of the command is then presented to you, see [Figure 106 on page 1039](#)

```
***** 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 CSQMCA ' 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 106. 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 107. 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 108. 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)
Object type	SYSTEM
Name	Blank

The Start a System Function panel is displayed (see [Figure 107 on page 1040](#)).

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 108 on page 1041](#)).

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 109. 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 110. 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 110 on page 1044](#)).

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 110 on page 1044](#)).

Select Function type 3 or 4 (resolve with commit or backout). (See [“Handling in-doubt channels” on page 254](#) 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 111. 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 252](#) for more information. For information about restarting stopped channels, see [“Restarting stopped channels” on page 253](#).

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 104 on page 1036](#)), a summary of the channel status is shown for each channel as follows:

INACTIVE	No connections are active
<i>status</i>	One connection is active
<i>nnn status</i>	More than one connection is current and all current connections have the same status
<i>nnn CURRENT</i>	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 112](#) on [page 1047](#)).

```
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 112. 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 244](#) 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 113 on page 1048](#), 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      -      MQ25      N
-  MQ90
-  TO.MQ95.T      HURSLEY.MACH95.COM(1595)      RUN
-  CLUSSDRA      VJH01T      -      MQ25      N
-  MQ95
-  TO.MQ96.T      HURSLEY.MACH96.COM(1596)      RUN
-  CLUSSDRB      VJH01T      -      MQ25      N
-  MQ96
***** 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 113. 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).

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 2.4: UJ00636
- z/OS 2.3: UJ00635
- z/OS 2.2: UJ00638
- z/OS 2.1: UJ00639

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.

z/OS 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 1050](#)
- [“Defining an LU6.2 connection for z/OS using APPC/MVS” on page 1053](#)

A message channel that uses TCP/IP can be pointed at an IBM Aspera faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. A queue manager running on any entitled platform can connect through an Aspera gateway. The gateway itself is deployed on Red Hat or Ubuntu Linux, or Windows. See [Defining an Aspera gateway connection on Linux or Windows](#).

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 1058](#).

Related concepts

[“Using the panels and the commands” on page 1035](#)

You can use the MQSC commands, the PCF commands, or the operations and control panels to manage DQM.

[“Setting up IBM MQ for z/OS” on page 957](#)

Use this topic as a step-by-step guide for customizing your IBM MQ for z/OS system.

[“Monitoring and controlling channels on z/OS” on page 1033](#)

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 1054](#)

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 1058](#)

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.

Related tasks

[“Configuring distributed queuing” on page 212](#)

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 z/OS” on page 1030](#)

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed queuing.

z/OS 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
- /etc/resolv.conf on the file system
- //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 SYS1.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.

 A message channel that uses TCP/IP can be pointed at an IBM Aspera faspio Gateway, which provides a fast TCP/IP tunnel that can significantly increase network throughput. A queue manager running on any entitled platform can connect through an Aspera gateway. The gateway itself is deployed on Red Hat or Ubuntu Linux, or Windows. See [Defining an Aspera gateway connection on Linux or Windows](#).

Related concepts

[“Sending end” on page 1052](#)

At the sending end of the TCP/IP connection, there are a number of settings to configure.

[“Receiving on TCP” on page 1052](#)

At the receiving end of the TCP/IP connection, there are a number of settings to configure.

[“Using the TCP listener backlog option on z/OS” on page 1052](#)

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 (CONNNAME) 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 on z/OS

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 10000. 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.

Related concepts

[“Using the TCP listener backlog option on IBM MQ for Multiplatforms” on page 284](#)

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.

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:

<i>Table 84. Settings on the local z/OS system for a remote queue manager platform</i>	
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.
AIX 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.

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 1054](#)

To connect to LU 6.2, there are a number of settings to configure.

[“Receiving on LU 6.2” on page 1054](#)

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 (CONNNAME) 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 1058](#).

For information about how to monitor and control channels when using queue sharing groups, see [“Monitoring and controlling channels on z/OS” on page 1033](#).

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 1056](#).

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 1056](#).

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](#)

[“Setting up IBM MQ for z/OS” on page 957](#)

Use this topic as a step-by-step guide for customizing your IBM MQ for z/OS system.

[“Clusters and queue sharing groups” on page 1058](#)

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 1058](#)

During shared queue peer recovery, message channel agents that process messages on shared queues serialize their access to the queues.

Intra-group queuing

Related tasks

[“Configuring distributed queuing” on page 212](#)

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 z/OS” on page 1030](#)

This section describes the IBM MQ for z/OS preparations you need to make before you can start to use distributed 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 1053](#).

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](#)

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 1054](#)) that is triggered to start channel QSG.T0.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.T0.QM2)
```

Note: If a shared queue is setup for triggering and connection to the Coupling Facility hosting the shared queue is lost, a trigger event might be generated and a message put to the initiation queue. This can happen even when no message was put to the original shared queue setup for triggering. This is caused by the over-indication of bits by the IXLVECTR macro as documented in [The List Notification Vector](#).

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 z/OS” on page 1032](#)). 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 1054](#).

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 309](#).

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 1050](#).

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 (CONNNAME) 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 1060](#)
- [“Setting up the IMS bridge” on page 1066](#)

Related concepts

[IBM MQ and IMS](#)

[“Using IBM MQ with CICS” on page 1067](#)

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 1069](#)

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 tasks

[“Configuring queue managers on z/OS” on page 952](#)

Use these instructions to configure queue managers on IBM MQ for z/OS.

Related reference

[“Upgrading and applying service to Language Environment or z/OS Callable Services” on page 1067](#)

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 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 1061](#).

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 for your national language:

- C, for Chinese
- E, for US English mixed case
- F, for French
- K, for Japanese
- U, for US English uppercase

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 1064](#) 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 114 on page 1061](#).
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 1065](#).

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(CSQSTUB)
INCLUDE IMSLIB(DFSII000)
ALIAS MQCONN,MQCONN,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,MQMHBUFF,MQBUFMH IMS entry points
MODE AMODE(31),RMODE(24) Note RMODE setting
NAME CSQQDYN(R)
/*

1Specify the name of a library accessible to IMS applications that
want to make dynamic calls to IBM MQ.
```

Figure 114. Sample JCL to link-edit the dynamic call stub

Related concepts

IBM MQ and IMS

“Setting up the IMS bridge” on page 1066

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. The SST parameter can have a value of either DB2 or MQ. Support for the MQ value was added in IMS 14. Use of MQ aids clarity, and the IMS subsystem command now includes the SST value, but otherwise does not have any significant effect. A value of DB2 can still be used if required. Other parameters are as described in [Positional parameters](#), and shown in the following example:

```
SST=MQ,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 85 on page 1063](#) shows the different possibilities of SSM specifications.

Table 85. 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 85. 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.

▶ z/OS 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 115 on page 1064](#) shows the syntax of this assembler macro.

```
CSQQDEFX TYPE=ENTRY|DEFAULT,NAME=qmgr-name,LIT=token
or
CSQQDEFX TYPE=END
```

Figure 115. 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 MQCONNEX 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 **MQCONNEX**.

LIT= token

Specifies the name of the language interface token (LIT) that IMS uses to identify the queue manager.

An MQCONN or MQCONNEX 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 MQCONNEX 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 MQCONNEX 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 MQCONNEX call, see [MQCONNEX - 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 116 on page 1065](#) 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 116. 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 117. 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 118. 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.

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 1060](#)

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](#)

▶ **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 IBM MQ](#) section of the CICS documentation.

Related concepts

[IBM MQ and CICS](#)

[“Using IBM MQ with IMS” on page 1059](#)

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 1067](#)

The actions you must take vary according to whether you use CALLLIBS or LINK, and your version of SMP/E.

▶ **z/OS**

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 86. Service has been applied or the product has been upgraded to a new release		
Product	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.	Action if using LINK
Language Environment	1. Set the Boundary on your SMP/E job to the Target zone. 2. On the SMP_CNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See z/OS SMP/E Commands for further information. 3. Run the SMP/E job.	No action required provided that the SMP/E zones were set up for automatic relinking, and the CSQ8SLDQ job has been run.
Callable Services	1. Set the Boundary on your SMP/E job to the Target zone. 2. On the SMP_CNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See z/OS SMP/E Commands for further information. 3. Run the SMP/E job.	No action required provided that the SMP/E zones were set up for automatic relinking, and the CSQ8SLDQ job has been run.

Table 87. One of the products has been updated to a new release in a new SMP/E environment and libraries		
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
Language Environment	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 SMP_CNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See z/OS SMP/E 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: CMQXDCST, CMQXRCTL, CMQXSUPR, CSQCBE00, CSQCBE30, CSQCBP00, CSQCBP10, CSQCBR00, CSQUCVX, CSQUDLQH, CSQVXPCB, CSQVXSPT, CSQXDCST, CSQXRCTL, CSQXSUPR, CSQXTDMI, 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.

Table 87. 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	Action if using LINK
	Note: You do not need to run three separate jobs for Language Environment and Callable services. One job will suffice for both products.	
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 SMP_CNTL card specify LINK LMODS CALLLIBS. You can also specify other parameters such as CHECK, RETRY(YES) and RC. See <i>z/OS SMP/E Commands</i> 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 1069.

▶ z/OS 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 119. Example SMP/E LINK CALLLIBS job

▶ z/OS 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. See [DFSYDRU0](#) for more information.

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 routing exit for IMS:

- [“The pre-routing exit DFSYPRX0” on page 1070](#)
- [“The destination resolution user exit” on page 1071](#)

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 1059](#)

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. See [OTMA Destination Resolution user exit \(DFSYPRX0 and other OTMAYPRX type exits\)](#) for parameters passed to this routine by IMS.

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. You should 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 120 on page 1071:

```

TITLE 'DFSYPX0: OTMA PRE-ROUTING USER EXIT'
DFSYPX0 CSECT
DFSYPX0 AMODE 31
DFSYPX0 RMODE ANY
*
SAVE (14,12),,DFSYPX0&SYSDATE&SYSTIME
SPACE 2
LR R12,R15          MODULE ADDRESSABILITY
USING DFSYPX0,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 120. 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 DFSYPX0, 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.

The DRU exit should be 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. See [Specifying synchronized tpipes for IBM MQ](#) for more information.

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 121 on page 1073](#).

```

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
*
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
*
B EXIT0             SET REPLYTO FORMAT NAME
*
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 121. Sample assembler DRU exit

Enabling MFT agents to connect to remote z/OS queue managers

Subject to entitlement, an MFT agent on z/OS can use a client connection to connect to a z/OS queue manager. This can lead to simpler IBM MQ topologies.

If an MFT agent on z/OS is associated with the product identifier (PID) of either IBM MQ Advanced for z/OS VUE or IBM MQ Advanced for z/OS, then the agent can use a client connection to connect to a queue manager on z/OS.

For information on the different PIDs, see [IBM MQ product identifiers and export information](#). For information on how to set the PID associated with an MFT installation, see [fteSetProductId](#).

The PID under which the agent is running is displayed in the log at agent startup.

An MFT agent on z/OS, running under any other PID, can connect only to a local queue manager using a bindings mode connection.

If an agent attempts to connect to a queue manager that is not running on z/OS, the BFGQM1044E message is issued, and agent start-up is ended.

Related tasks

[Starting an MFT agent on z/OS](#)

[Stopping an MFT agent on z/OS](#)

Related reference

[The MFT agent.properties file](#)

Configuring IBM MQ Internet Pass-Thru

This section describes the various features that IBM MQ Internet Pass-Thru (MQIPT) supports, and how to configure them.

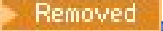
Configure MQIPT by making changes to the configuration file `mqipt.conf`. The structure of the MQIPT configuration file and the properties that can be specified are described in [IBM MQ Internet Pass-Thru configuration reference](#).

Note: You should set secure file permissions on the directory where the `mqipt.conf` file is located to prevent unauthorized users seeing any stored passwords or changing the configuration. Protect all passwords specified in the configuration file by following the procedure in [“Encrypting stored passwords in MQIPT”](#) on page 1108.

Changes to the configuration file take effect when MQIPT is started or refreshed. Refreshing an active instance of MQIPT brings configuration changes into effect without restarting MQIPT. When MQIPT is refreshed, the `mqipt.conf` configuration file is re-read and MQIPT takes the following actions:

- Any active routes that are marked as inactive, or are no longer specified in the configuration file, are closed and no longer accept incoming connections.
- Any routes that are marked as active in the configuration file, and are not currently running, are started.
- Any changes to the configuration parameters of active routes are applied. Where possible, these changes take effect without any disruption to active connections. For some parameter changes, such as a change to the route destination, all connections are closed before the change is applied and the route is restarted.

To refresh MQIPT, use the **mqiptAdmin** command. For more information on administering MQIPT using the **mqiptAdmin** command, see [Administering MQIPT by using the command line](#).

 Support for the Java security manager, which was previously deprecated, is removed at IBM MQ 10.0.0. The **SecurityManager** and **SecurityManagerPolicy** properties are both removed. If you are migrating to IBM MQ 10.0.0 from an earlier version that has **SecurityManager** configured, you should remove the **SecurityManager** and **SecurityManagerPolicy** properties from the MQIPT configuration to allow MQIPT to start up successfully. If you do not do this, MQIPT will fail to start with the following messages:

```
MQCPE126 Global property SecurityManager is invalid.
MQCPE126 Global property SecurityManagerPolicy is invalid.
MQCPI002 IBM MQ Internet Pass-Thru 10.0.0.0 shutting down
MQCPI003 IBM MQ Internet Pass-Thru 10.0.0.0 shutdown complete
```

HTTP support in MQIPT

MQIPT supports HTTP tunneling. MQIPT can be configured so that the data packets it forwards are encoded as HTTP requests.

IBM MQ channels do not accept HTTP requests. Therefore a second MQIPT is required to receive the HTTP requests and convert them back into IBM MQ protocol packets. The second MQIPT removes the HTTP header to convert the incoming packet back into a standard IBM MQ protocol packet, before passing it on to the destination queue manager.

Note: From IBM MQ 9.4.0, MQIPT routes do not accept HTTP connections by default. Routes must be configured to accept HTTP connections by using the **AllowedProtocols** property.

When HTTP is being used between two instances of MQIPT, the TCP/IP connection on which the HTTP requests and replies flow is persistent and is kept open for the lifetime of the message channel. MQIPT does not close the TCP/IP connection between request/reply pairs.

If two instances of MQIPT are communicating through HTTP, it is possible that an HTTP request might stay outstanding for an extended period. An example is in a requester/server channel, when the server side is waiting for new messages to arrive on its transmission queue. The IBM MQ channel protocol provides a "heartbeat" mechanism, which requires the waiting end periodically to send heartbeat messages to its partner. The default channel heartbeat period is 5 minutes. MQIPT uses this heartbeat as the HTTP reply. Do not disable this channel heartbeat, or set it to an excessively high value, to avoid causing problems with timeouts in some firewalls.

MQIPT accepts HTTP traffic in chunked format, generated by an HTTP proxy or server.

For an example of using HTTP in MQIPT, see [Configuring HTTP tunneling](#).

HTTP proxies

A HTTP proxy can be placed in between the two instances of MQIPT. The HTTP proxy must meet the following requirements:

- The proxy must support the HTTP 1.1 protocol.
- The **Connection** or **Proxy-Connection** HTTP headers that are set by MQIPT must be honored by the proxy. This allows connections between the two instances of MQIPT to be kept open for the lifetime of the message channel.
- A one-to-one mapping of persistent connections must be maintained across the proxy. This ensures that TCP/IP connections from the proxy to the destination MQIPT are not used to transmit data for more than one message channel.

You can set properties to configure how persistent connections are managed on some HTTP proxies. For example, you might be able to set the maximum number of requests that can be made on a persistent connection. The following properties should be set:

- Persistent connections should be enabled.
- The reuse of TCP/IP connections from the proxy to MQIPT by more than one HTTP session should be disabled, to maintain a one-to-one mapping of persistent connections across the proxy.
- The timeout on proxy requests should be set to a high value. For example, 12 hours.
- The maximum number of requests that can be made on a persistent connection should be set to a high value. For example, 5000.

MQIPT uses HTTP POST requests to send data between the two instances of MQIPT. If the MQIPT configuration specifies the hostname of the proxy using the **HTTPProxy** property, MQIPT connects to the proxy and uses the HTTP CONNECT method to request that the proxy establish a tunnel to the destination MQIPT. This allows HTTPS connections to pass through the proxy without terminating the TLS session in the proxy.

If a load balancer is placed between the MQIPT instances, it must be configured to use the value of the *MQIPTSessionId* HTTP cookie to ensure that all requests for each session are forwarded to the same destination.

HTTPS in MQIPT

HTTPS can be used on an HTTP connection by enabling the **HTTPS** and **SSLClient** route properties on the MQIPT issuing the client connection.

MQIPT must have access to the trusted CA certificate that will be used to authenticate the target HTTP proxy/server. The **SSLClientCAKeyring** property can be used to define the key ring file containing the trusted CA certificate.

A common setup for HTTPS will use a local HTTP proxy to tunnel out through a firewall and connect to a remote HTTP server (or another proxy), which will in turn connect to the remote MQIPT. This MQIPT on

the server side of the connection does not need any specific configuration, as the connection request is treated as any normal HTTP connection.

MQIPT uses the **HTTPProxy** and **HTTPServer** properties to distinguish the local and remote proxies. The MQIPT route with the **HTTPProxy** property set is seen to be the local HTTP proxy, and the MQIPT route with the **HTTPServer** property set is the remote server (or proxy).

HTTPS connections are normally made to listener port address 443 on the HTTP proxy/server, but the **HTTPProxyPort** and **HTTPServerPort** properties can be used to override this default.

SOCKS support in MQIPT

A SOCKS proxy is a network service used as a controlled point of exit through a firewall. A SOCKS enabled application, running inside the firewall, can use the SOCKS proxy to connect to a remote application.

MQIPT can act as a SOCKS proxy by enabling the **SocksServer** property, thereby allowing a SOCKS-enabled IBM MQ application to connect through MQIPT to a remote IBM MQ queue manager. When using this feature, the target destination and destination port address are obtained during the SOCKS handshaking process and therefore the **Destination** and **DestinationPort** route properties are overridden. This is a key feature for supporting IBM MQ clustering.

MQIPT can also act as a SOCKS client, on behalf of a local IBM MQ application which has not been SOCKS enabled. This is useful when using a firewall that allows outbound connections only via a SOCKS proxy. Each MQIPT route can be configured to communicate with a different SOCKS proxy.

See [Configuring a SOCKS proxy](#) for an example of how to use SOCKS.

Clustering in MQIPT

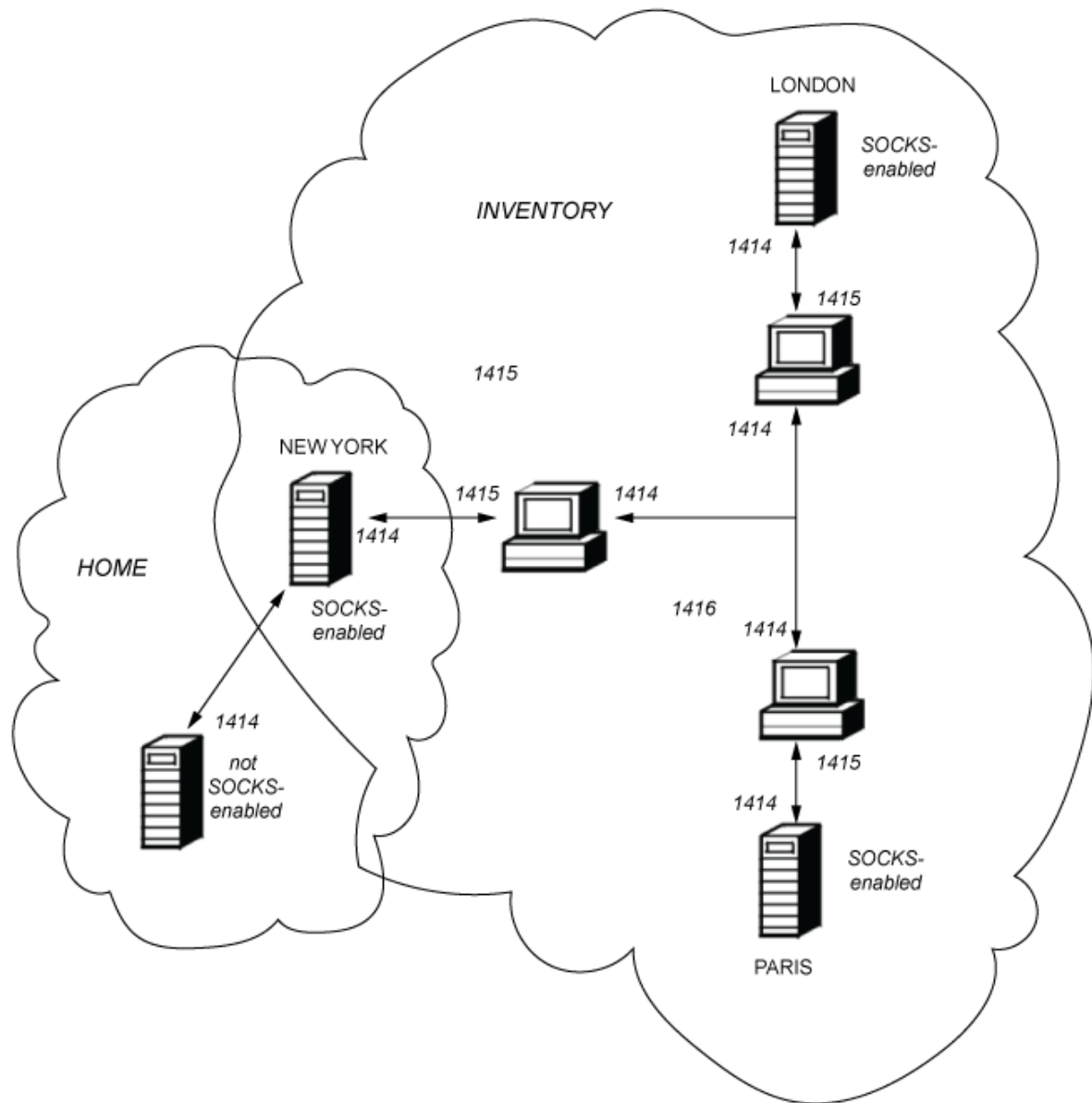
IBM MQ clusters can be used with MQIPT by SOCKS-enabling each queue manager in the cluster that spans the internet and by enabling MQIPT to act as SOCKS proxy.

In the following diagram, NEWYORK and CHICAGO are in a cluster called HOME and both hold full repositories. NEWYORK, LONDON and PARIS are in another cluster called INVENTORY. Note that CHICAGO does not need to be SOCKS-enabled as it is in a cluster that does not need an MQIPT.

Each queue manager in the INVENTORY cluster is effectively "hidden" behind an MQIPT. As the queue manager has been SOCKS-enabled, when a cluster-sender channel is started, the request is sent out to its destination, using MQIPT acting as a SOCKS proxy. Normally, the CONNAME on a cluster-receiver channel is used to identify the local queue manager, but when used with MQIPT, the CONNAME must identify the local MQIPT and its incoming listener port. In the following diagram, all incoming listener port addresses are 1414 and the outgoing listener port addresses are 1415.

There are two ways to run a SOCKS-enabled queue manager. The first is to SOCKS-enable the whole computer where the queue manager is running. The second is to SOCKS-enable just the queue manager. Using either method, you must configure the SOCKS client so it only makes remote connections using MQIPT as the SOCKS proxy and disable user authentication. There are a number of products on the market to achieve SOCKS support. You must choose one that supports SOCKS V5 protocol.

See [Configuring MQIPT clustering support](#) for an example of how to configure a cluster network.



SSL/TLS support in MQIPT

Secure sockets can be used to ensure communication privacy, communication integrity, and authentication.

Communication privacy

The connection can be made private. The data to be exchanged between the client and the server can be encrypted and only the sender and receiver can make sense of the data. This means that private information, such as credit card numbers, can be transferred securely.

Communication integrity

The connection is reliable. The message transport includes a message integrity check based on a secure hash function.

Authentication

The client can authenticate the server and an authenticated server can authenticate the client. This means that the information is guaranteed to be exchanged only between the intended parties. The authentication mechanism is based on the exchange of digital certificates (X.509v3 certificates).

Secure sockets protocols

MQIPT supports the following secure sockets protocols:

- SSL 3.0
- TLS 1.0
- TLS 1.1
- TLS 1.2
- TLS 1.3

SSL 3.0, TLS 1.0, and TLS 1.1 are insecure and are disabled by default in MQIPT. If you need to use any of these disabled protocols, they can be re-enabled by following the procedure in [“Enabling deprecated protocols and Cipher Suites in MQIPT”](#) on page 1099.

The combination of the IBM MQ CipherSpec of the remote channel and the protocol in the MQIPT route definition determines which protocol is used by a connection to or from MQIPT.

The SSL/TLS protocols can use different digital signature algorithms for authentication of communication parties. The cryptographic operations that are used in SSL/TLS, encryption for data confidentiality, and secure hashing for message integrity, rely on the sharing of secret keys between the client and the server. SSL/TLS provides various key exchange mechanisms that allow for the sharing of secret keys. SSL/TLS can use various algorithms for encryption and hashing.

FIPS support in MQIPT

 From IBM MQ 9.4.4, MQIPT is supplied with a IBM Semeru Runtime 21 Java runtime environment which supports FIPS 140-3 cryptography.

Note: From IBM MQ 9.4.4, FIPS is not supported for MQIPT on Linux s390.

 If you want to use only FIPS certified cryptography in MQIPT, specify the **-sm** parameter on the **mqipt**, **mqiptAdmin**, and **mqiptPW** commands.

SSL/TLS Bridging Mode

When a route has both **SSLServer** and **SSLClient** set, MQIPT accepts one incoming SSL/TLS secured connection and establishes a second SSL/TLS secured connection to another MQIPT or to a destination queue manager. The IBM MQ channel information is decrypted and re-encrypted between these two SSL/TLS connections. SSL/TLS bridging is also referred to as an *SSL/TLS termination proxy*.

IBM MQ supports SSL/TLS bridging by using MQIPT. Other SSL/TLS termination proxies have been observed to cause broken connections if the proxy combines or reconstructs SSL/TLS records with different sizes than those sent by IBM MQ. This is due to an interaction between the way queue managers allocate and manage memory for incoming IBM MQ network data and the way IBM MQ network data is packaged into SSL/TLS records.

MQIPT preserves the packaging of IBM MQ network data in SSL/TLS records without splitting or combining them. If other SSL/TLS bridges do not preserve the SSL/TLS records exactly, they can cause IBM MQ channels to fail with one of the following error messages:

- AMQ9638: SSL communications error for channel
- AMQ9208: Error on receive from host

SSL/TLS Proxy Mode

An MQIPT route can be configured in SSL/TLS proxy mode as an alternative to SSL/TLS bridging. In this mode, the route only forwards SSL/TLS data between two IBM MQ end-points; it does not participate in the SSL/TLS handshake and does not require any digital certificates.

You can use SSL/TLS proxy mode in cases where the IBM MQ channels which communicate through MQIPT are already configured for SSL/TLS communication and you want to use MQIPT for another

purpose, such as routing connections through firewalls or restricting the set of allowable connections through a security exit. When running in SSL/TLS proxy mode, MQIPT checks that the initial SSL/TLS packets received from a new connection are valid before forwarding the packets to the destination.

IBM MQ supports SSL/TLS proxy mode for connections that pass through MQIPT or any other SSL/TLS proxy.

IBM MQ multiple certificate support with MQIPT

IBM MQ supports the use of multiple certificates on the same queue manager, using a per-channel certificate label that is specified by using the **CERTLABL** attribute on the channel definition. Inbound channels to the queue manager detect the channel name by using TLS Server Name Indication (SNI), in order to present the correct certificate from the queue manager. For more information about using multiple certificates on a queue manager, see [How IBM MQ provides multiple certificates capability](#).

If a channel connects to the destination queue manager through MQIPT, and the MQIPT route is configured in TLS bridging mode, there are two separate TLS sessions between the endpoints. The SNI data that is sent by MQIPT when it connects to the route destination can be configured by using the **SSLClientOutboundSNI** route property. To allow per-channel certificates to be used on a destination queue manager, the route must either be configured to set the SNI to the IBM MQ channel name, or to pass through the SNI that is received on the inbound connection to the route. If MQIPT is configured to pass through the SNI, the queue manager or client that connects to MQIPT must set the SNI to the channel name.

The certificates that are used for TLS connections that are terminated or initiated by MQIPT can be configured individually for each route, for example by using the **SSLServerSiteLabel** and **SSLClientSiteLabel** route properties.

CipherSuites supported by MQIPT from IBM MQ 10.0.0

MQIPT routes that use TLS must be configured to use a CipherSuite that corresponds to the CipherSpec that is used by the IBM MQ channel. From IBM MQ 10.0.0 (or IBM MQ 9.4.4 for Continuous Delivery), you must ensure that your MQIPT configuration specifies CipherSuites that are supported by IBM Semeru Runtime 25.

  From IBM MQ 10.0.0, the Java runtime environment (JRE) shipped with the product is updated from IBM Semeru Runtime 21 JRE to IBM Semeru Runtime 25. IBM Semeru Runtime 25 does not support FIPS 140-2 cryptography. If you are using any FIPS 140-2 CipherSuites, you will need to move to using FIPS 140-3 CipherSuites.

Table 88 on page 1079 shows which CipherSuites MQIPT supports from IBM MQ 10.0.0, and which are enabled by default.

By default, only a subset of CipherSuites is enabled. CipherSuites that are based on several algorithms that are considered insecure are disabled by the Java runtime environment (JRE). If you are aware of the potential hazards but still need to use one of these CipherSuites, you can add support for a disabled CipherSuite by following the procedure in [“Enabling deprecated protocols and Cipher Suites in MQIPT”](#) on page 1099.

Table 88. CipherSuites that you can use with MQIPT	
CipherSuite	Enabled by default
CipherSuites for TLS 1.3	
TLS_AES_128_GCM_SHA256	Yes
TLS_AES_256_GCM_SHA384	Yes
TLS_CHACHA20_POLY1305_SHA256	Yes
CipherSuites for SSL 3.0, TLS 1.0, TLS 1.1 and TLS 1.2	

Table 88. CipherSuites that you can use with MQIPT (continued)

CipherSuite	Enabled by default
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	
SSL_DHE_DSS_WITH_DES_CBC_SHA	
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	
SSL_DHE_RSA_WITH_DES_CBC_SHA	
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA	
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5	
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	
SSL_DH_anon_WITH_DES_CBC_SHA	
SSL_DH_anon_WITH_RC4_128_MD5	
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA	
SSL_RSA_EXPORT_WITH_RC4_40_MD5	
SSL_RSA_WITH_3DES_EDE_CBC_SHA	
SSL_RSA_WITH_DES_CBC_SHA	
SSL_RSA_WITH_NULL_MD5	
SSL_RSA_WITH_NULL_SHA	
SSL_RSA_WITH_RC4_128_MD5	
SSL_RSA_WITH_RC4_128_SHA	
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	Yes
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	Yes
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	Yes
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	Yes
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	Yes
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	Yes
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	Yes
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	Yes
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	Yes
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	Yes
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	Yes
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	Yes
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	Yes
TLS_DH_anon_WITH_AES_128_CBC_SHA	
TLS_DH_anon_WITH_AES_128_CBC_SHA256	

Table 88. CipherSuites that you can use with MQIPT (continued)

CipherSuite	Enabled by default
TLS_DH_anon_WITH_AES_128_GCM_SHA256	
TLS_DH_anon_WITH_AES_256_CBC_SHA	
TLS_DH_anon_WITH_AES_256_CBC_SHA256	
TLS_DH_anon_WITH_AES_256_GCM_SHA384	
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	Yes
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	Yes
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	Yes
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	Yes
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	Yes
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	Yes
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	Yes
TLS_ECDHE_ECDSA_WITH_NULL_SHA	
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Yes
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	Yes
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Yes
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Yes
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	Yes
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Yes
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	Yes
TLS_ECDHE_RSA_WITH_NULL_SHA	
TLS_ECDHE_RSA_WITH_RC4_128_SHA	
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA	
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	Yes
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	Yes
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	Yes
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	Yes
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	Yes
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	Yes
TLS_ECDH_ECDSA_WITH_NULL_SHA	
TLS_ECDH_ECDSA_WITH_RC4_128_SHA	
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA	

Table 88. CipherSuites that you can use with MQIPT (continued)

CipherSuite	Enabled by default
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	Yes
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	Yes
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	Yes
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	Yes
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	Yes
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	Yes
TLS_ECDH_RSA_WITH_NULL_SHA	
TLS_ECDH_RSA_WITH_RC4_128_SHA	
TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA	
TLS_ECDH_anon_WITH_AES_128_CBC_SHA	
TLS_ECDH_anon_WITH_AES_256_CBC_SHA	
TLS_ECDH_anon_WITH_NULL_SHA	
TLS_ECDH_anon_WITH_RC4_128_SHA	
TLS_RSA_WITH_AES_128_CBC_SHA “1” on page 1082	Yes
TLS_RSA_WITH_AES_128_CBC_SHA256 “1” on page 1082	Yes
TLS_RSA_WITH_AES_128_GCM_SHA256 “1” on page 1082	Yes
TLS_RSA_WITH_AES_256_CBC_SHA “1” on page 1082	Yes
TLS_RSA_WITH_AES_256_CBC_SHA256 “1” on page 1082	Yes
TLS_RSA_WITH_AES_256_GCM_SHA384 “1” on page 1082	Yes
TLS_RSA_WITH_NULL_SHA256 “1” on page 1082	

Notes:

1. The IBM Java 8 JRE removed support for RSA key exchange when operating in FIPS mode. TLS_RSA_* CipherSpecs are disabled by default in IBM Semeru Runtime 21 and IBM Semeru Runtime 25.

CipherSpecs and MQIPT CipherSuites

Table 89 on page 1083 shows the relationship between the CipherSpecs that are supported by IBM MQ and the CipherSuites that are supported by MQIPT. The table also shows the protocol version that IBM MQ expects each CipherSpec to use.

When an IBM MQ queue manager or client communicates with MQIPT, unless you are using SSL proxy mode in MQIPT, ensure that the CipherSpec that is used by IBM MQ matches the CipherSuite that is used by MQIPT:

- When MQIPT acts as the TLS server and IBM MQ connects as the TLS client, the CipherSpec that is used by IBM MQ must correspond to a CipherSuite that is selected in the MQIPT route configuration.
- When MQIPT acts as the TLS client and connects to an IBM MQ queue manager that acts as the TLS server, the MQIPT CipherSuite must match the CipherSpec that is defined on the receiving IBM MQ channel.

An IBM MQ CipherSpec determines both the encryption algorithm and the secure socket protocol version that is used. To use a specific CipherSuite for a connection between an IBM MQ client or queue manager

and MQIPT, specify both the CipherSuite and the protocol in the MQIPT route definition. These values must correspond to the CipherSpec that is specified in the IBM MQ channel definition. Specifying both the CipherSuite and the protocol in the MQIPT route definition avoids ambiguity as some CipherSuites are supported by more than one protocol.

For example, on a TLS client route, set both the **SSLClientCipherSuites** and **SSLClientProtocols** route properties to values that correspond to the intended CipherSpec. Use the protocol version in the table to determine the correct values for these route properties.

If you use a single MQIPT route to tunnel several IBM MQ channels that use different CipherSpecs, ensure that all channels are configured with a CipherSpec that uses the same secure sockets protocol version and that you set the **SSLClientProtocols** route property to this protocol.

For more information about IBM MQ CipherSpecs, see [Enabling CipherSpecs](#).

<i>Table 89. IBM MQ CipherSpecs that correspond to MQIPT CipherSuites</i>		
IBM MQ CipherSpec	MQIPT CipherSuite	Protocol version
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA	SSLv3
DES_SHA_EXPORT1024	N/A	N/A
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLSv1.2
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLSv1.2
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLSv1.2
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLSv1.2
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLSv1.2
ECDHE_ECDSA_NULL_SHA256	TLS_ECDHE_ECDSA_WITH_NULL_SHA	TLSv1.2
ECDHE_ECDSA_RC4_128_SHA256	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLSv1.2
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1.2
ECDHE_RSA_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLSv1.2
ECDHE_RSA_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLSv1.2
ECDHE_RSA_AES_256_CBC_SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLSv1.2
ECDHE_RSA_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLSv1.2
ECDHE_RSA_NULL_SHA256	TLS_ECDHE_RSA_WITH_NULL_SHA	TLSv1.2
ECDHE_RSA_RC4_128_SHA256	TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLSv1.2
NULL_MD5	SSL_RSA_WITH_NULL_MD5	SSLv3
NULL_SHA	SSL_RSA_WITH_NULL_SHA	SSLv3
RC2_MD5_EXPORT	N/A	N/A
RC4_56_SHA_EXPORT1024	N/A	N/A
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5	SSLv3
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	SSLv3
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA	SSLv3

Table 89. IBM MQ CipherSpecs that correspond to MQIPT CipherSuites (continued)

IBM MQ CipherSpec	MQIPT CipherSuite	Protocol version
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256	TLSv1.3
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384	TLSv1.3
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256	TLSv1.3
TLS_RSA_WITH_3DES_EDE_CBC_SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLSv1.2
TLS_RSA_WITH_AES_128_GCM_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLSv1.2
TLS_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLSv1.2
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384	TLSv1.2
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	TLSv1
TLS_RSA_WITH_NULL_NULL	N/A	N/A
TLS_RSA_WITH_NULL_SHA256	TLS_RSA_WITH_NULL_SHA256	TLSv1.2
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	TLSv1.2
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA	SSLv3

MQIPT and alias CipherSpecs

For ease of configuration and future migration, IBM MQ also provides a set of alias CipherSpecs. Using an alias CipherSpec means that the TLS handshake negotiates the best compatible ciphers for both ends of the channel. You can therefore adapt to cipher additions and deprecations without needing to make further configuration changes.

To use an alias CipherSpec with a channel that passes through MQIPT, specify only the protocols that you want to use in the MQIPT route definition, and do not specify the CipherSuite.

For example, if a sender channel is configured to use the alias CipherSpec ANY_TLS13, and the channel connects to an MQIPT route that is defined with `SSLServer=true`, specify the following property in the MQIPT route definition:

```
SSLServerProtocols=TLSv1.3
```

Do not specify the **SSLServerCipherSuites** property in the route definition. This allows the TLS handshake that involves MQIPT to negotiate the best CipherSuite, that is supported by both MQIPT and IBM MQ, from the set of protocols that are specified in the configuration.

The protocols that are specified in the MQIPT route definition must be compatible with the protocols that are enabled by the alias CipherSpec in the IBM MQ channel definition. For more information about connections that use alias CipherSpecs, see [Relationship between alias CipherSpec settings](#).

SSL/TLS handshake in MQIPT

The SSL/TLS handshaking process occurs during the initial connection request between the SSL/TLS client and server, when authentication and negotiation of CipherSuites is performed.

All the supported SSL/TLS CipherSuites (see “SSL/TLS support in MQIPT” on page 1077), with the exception of the anonymous CipherSuites, require server authentication and allow client authentication;

the server can be configured to request client authentication. You should avoid using anonymous CipherSuites because they provide no guarantees about the identity of the remote peer. It is possible for a man-in-the-middle attack to intercept anonymous SSL/TLS connections without your knowledge. Use anonymous CipherSuites only on trustworthy internal networks and only if you are prepared to accept the risk of data interception.

The communication peer authentication in SSL/TLS is based on public key cryptography and X.509v3 digital certificates. A site that should be authenticated in the SSL/TLS protocol requires a private key and a digital certificate (which contains the corresponding public key together with the information about the site's identity), validity time of the certificate. The certificates are signed by a Certification Authority, the certificates of such authorities are called signer certificates. A certificate followed by one or more signer certificates constitute a certificate chain. A certificate chain is characterized by the fact that, starting from the first certificate (site certificate), the signature of each certificate in the chain can be verified using the public key contained in the next signer certificate.

When a secure connection requiring server authentication is being established the server sends to the client a certificate chain to prove its identity. The SSL/TLS client will pursue the connection establishment to the server only if it can authenticate the server, for example, verify the signature of the server's site certificate. In order to verify that signature the SSL/TLS client needs to trust the server site itself or at least one of the signers in the certificate chain provided by the server. The certificates of the trusted sites and signers must be maintained on the client side to perform this verification.

The SSL/TLS client inspects the server's certificate chain, starting with the site certificate. The client considers the signature of the site certificate to be valid in the following circumstances:

- The site certificate is in the repository of the trusted site or signer certificates
- A signer certificate in the chain can be validated based on its repository of trusted signer certificates

In the latter case, the SSL/TLS client checks that the certificate chain is indeed correctly signed, from the trusted signer certificate down to the server's site certificate. Each certificate involved in this process is also examined for correctness of format and dates of validity. If any of these checks fail, the connection to the server is refused. After verifying the server certificate the client uses the public key embedded in that certificate in the next steps of the SSL/TLS protocol. The SSL/TLS connection can only be established if the server really has the corresponding private key.

The client authentication follows the same procedure: if a SSL/TLS server requires client authentication the client sends to the server a certificate chain to prove its identity. The server verifies the chain based on its repository of trusted site and signer certificates. After verifying the client's certificate the server uses the public key embedded in that certificate in the next steps of the SSL/TLS protocol. The SSL/TLS connection can only be established if the client really has the corresponding private key.

Recent versions of the TLS protocols provide high security communications (SSL and older TLS protocols are considered insecure). However, the protocol operates based on the information provided by the application. Only if that information base is also maintained securely the overall goal of secure communication can be achieved. For example, if your repository of trusted site and signer certificates is compromised, you might establish a secure connection to a very insecure communication partner.

MQIPT implementation of SSL/TLS

SSL 3.0, TLS 1.0, TLS 1.1, TLS 1.2 and TLS 1.3 are implemented with Public Key Cryptography Standards (PKCS) #12 tokens stored in key ring files (with file types of .p12 or .pfx), containing X509.V3 certificates. MQIPT can also use cryptographic hardware key stores that support the PKCS#11 Cryptographic Token Interface standard. MQIPT uses the IBM Java Secure Socket Extension (JSSE) package.

MQIPT can act as an SSL/TLS client or an SSL/TLS server depending on which end initiates the connection. The client starts a connection and the server accepts the connection request. It is possible for an MQIPT route to act both as a client and a server. In this case, using the SSL/TLS Proxy Mode feature typically gives better performance.

When MQIPT is configured for SSL/TLS Proxy Mode, it only forwards SSL/TLS data between the two end-points; it does not participate in the SSL/TLS handshake and does not require any digital certificates.

In versions earlier than IBM MQ 9.3.0, MQIPT does not pass TLS Server Name Indication (SNI) data that is received on an inbound TLS connection through to an outbound TLS connection. This means that per-channel certificates, specified using the **CERTLABL** channel attribute, cannot be used for TLS connections between MQIPT and the destination queue manager. To use a per-channel certificate on the destination queue manager, for a TLS connection that passes through MQIPT at a version earlier than IBM MQ 9.3.0, the MQIPT route must use SSL/TLS Proxy Mode, which forwards all TLS control flows intact, including the SNI name. From IBM MQ 9.3.0, MQIPT can be configured to either set the SNI for TLS connections to a specific value, or to pass through the SNI received on the inbound connection to the route. For more information about using multiple certificates on a queue manager with MQIPT, see [“IBM MQ multiple certificate support with MQIPT” on page 1079](#).

Each MQIPT route can be independently configured with its own set of SSL/TLS properties. See [MQIPT route properties](#) for more details.

Encrypting a key ring password in MQIPT

Passwords that are used by MQIPT to open a key ring file, or to access cryptographic hardware, must be encrypted with the **mqiptPW** command.

The encrypted password can be specified in any of the following properties:

- **SSLClientKeyRingPW**
- **SSLClientCAKeyRingPW**
- **SSLServerKeyRingPW**
- **SSLServerCAKeyRingPW**
- **SSLCommandPortKeyRingPW**

In versions earlier than IBM MQ 9.1.5, key ring passwords for use by MQIPT are stored in files that the **SSL*KeyRingPW** properties reference.

Encrypt key ring passwords for use by MQIPT by using the **mqiptPW** command, and set the value of the **SSL*KeyRingPW** properties to the encrypted password. MQIPT can distinguish between encrypted passwords and file names in property values for compatibility with configurations created before IBM MQ 9.1.5.

 **Deprecated** The method of encrypting key ring passwords in MQIPT versions earlier than IBM MQ 9.1.5 is deprecated, but can still be used. To improve the protection of key ring passwords, re-encrypt any key ring passwords that are encrypted using the deprecated method, with the latest protection method.

To encrypt a key ring password for use by MQIPT, follow the steps in [“Encrypting stored passwords in MQIPT” on page 1108](#).

Selecting certificates from a key ring file in MQIPT

It is possible to have more than one personal certificate stored in the same key ring file or cryptographic hardware token. The **SSLClientSite*** properties can be used on the client side to select the certificate to be sent to the server for authentication and the **SSLServerSite*** properties can be used on the server side to select the certificate to be sent to the client for authentication.

Using these properties, a certificate can be selected based on its Distinguished Name (DN). Alternatively, the certificate label can be used to select a certificate using the **SSLServerSiteLabel1** and **SSLClientSiteLabel1** properties.

To select the server certificate used by the TLS command port, use the **SSLCommandPortSiteLabel1** property to specify the label name of the certificate.

Trust settings in MQIPT

A keystore contains a personal certificate that includes the signer certificate or chain of signer certificates.

MQIPT uses two types of keystores:

Certificate Authority (CA) keystore

This keystore contains trusted CA certificates that are used to validate certificates that belong to a remote peer. These CA certificates help to determine whether the remote peer is trustworthy. MQIPT supports both PKCS #12 format keystores, and cryptographic hardware keystores that support the PKCS #11 interface, for storing CA certificates.

The MQIPT CA keystores are identified by the **SSLClientCAKeyRing** and **SSLServerCAKeyRing** route properties. Use of cryptographic hardware to access CA certificates is enabled by setting the **SSLClientCAKeyRingUseCryptoHardware** and **SSLServerCAKeyRingUseCryptoHardware** properties.

The CA keystore on the SSL/TLS client side must contain the trusted CA certificates that are used to authenticate the certificate sent from the server. If a SSL server route is configured for client authentication, the CA keystore on the SSL/TLS server side must contain the trusted CA certificates that are used to authenticate the certificate sent from the client.

Personal certificate keystore

This keystore contains personal certificates that MQIPT uses to identify itself to a remote peer. When you generate a self-signed certificate or request a CA-signed certificate, do so by using the personal certificate keystore.

MQIPT supports both PKCS #12 format keystores, and cryptographic hardware keystores that support the PKCS #11 interface, for storing personal certificates.

Personal certificate keystores are identified by the **SSLClientKeyRing** and **SSLServerKeyRing** route properties. Use of cryptographic hardware to access personal certificates is enabled by setting the **SSLClientKeyRingUseCryptoHardware** and **SSLServerKeyRingUseCryptoHardware** properties.

The keystore on the SSL/TLS server side must contain the MQIPT server's personal certificate. If client authentication is needed on an SSL client route, the keystore on the SSL/TLS client side must contain the client's personal certificate.

If you need client authentication, you must enable the **SSLServerAskClientAuth** property on the server side, or set the **SSLServerClientAuth** property to required. The keystore on the client side must contain the client's personal certificate. The MQIPT keystore on the server side, which is identified by the **SSLServerCAKeyRing** property, must contain the trusted CA certificates that are used to authenticate the client.

If you do not configure a CA keystore for a route, MQIPT searches for CA certificates in the personal certificate keystore instead, if one is configured. For example, if no value is set for **SSLServerCAKeyRing**, MQIPT searches for CA certificates in the keystore that is identified by **SSLServerKeyRing**.

As an alternative to using certificates signed by a trusted CA, you can use self-signed certificates. You can find an example of a self-signed certificate in the `sslSample.pfx` sample keystore that is provided with MQIPT in the `samples/ssl` subdirectory. To open the sample PKCS#12 keystores, you must use the password `mqiptSample`.

Self-signed certificates can be useful in test scenarios where you must ensure SSL/TLS connectivity without paying a CA for a certificate. However, do not use self-signed certificates in production environments. To create a CA-signed certificate, see [Creating a key ring file](#).

You can use the **mqiptKeytool** command to manage digital certificates and keystores. For more information, see [“Managing MQIPT keystores” on page 1091](#).

Protect any keystores and password files by using the security features of the operating system to prevent unauthorized access to them.

Testing SSL/TLS in MQIPT

Examples to help you to test an SSL/TLS connection.

See [Getting started with IBM MQ Internet Pass-Thru](#) for a description of various scenarios. In particular, see the following tasks:

- [Authenticating an SSL/TLS server](#)
- [Authenticating an SSL/TLS client](#)
- [Running MQIPT in SSL/TLS proxy mode](#)
- [Running MQIPT in SSL/TLS proxy mode with a security manager](#)

To test that your SSL/TLS configuration works correctly, you can use self-signed certificates. Self-signed certificates are useful in test scenarios so that you can test SSL/TLS connectivity without paying a certificate authority (CA) for a certificate. For more information, see [Creating test certificates](#).

Do not use any self-signed certificates in production environments. Instead, obtain a CA-signed certificate from a trusted CA. To create a CA-signed certificate, see [Creating a key ring file](#).

When you create or request a certificate, you must consider which key type, key size, and digital signature algorithm are appropriate for your security needs. For more information, see [“Digital certificate considerations for MQIPT” on page 1092](#).

Certificates and certificate management technologies are available from several third-party suppliers.

MQIPT sample key rings

The following sample PKCS #12 key ring files are provided with MQIPT in the `samples/ssl` subdirectory for your convenience during testing:

sslSample.pfx

A key ring that contains a sample self-signed certificate.

sslCASample.pfx

A sample CA certificate key ring.

To access these sample key ring files, use the password `mqiptSample`.

The sample self-signed certificate must be used only in a test environment as the private keys of the certificate is available to all MQIPT users.

SSL/TLS error messages in MQIPT

Handshake failures are logged in the MQIPT connection log in the form of JSSE exceptions.

For more information, see [“Connection logs in MQIPT” on page 1111](#). The following table describes the different exceptions, the likely cause and the corresponding action to resolve the failure.

Certificate exceptions usually relate to the certificates at the remote end of the connection.

Where the error relates to the certificate of an IBM MQ client or queue manager, the term *key ring file* includes the IBM MQ key repository of the remote partner.

In MQIPT, CA certificates are stored in the CA key ring file, which is identified by the **SSLClientCAKeyRing** and **SSLServerCAKeyRing** route properties. If the CA key ring route properties are not set, the corresponding personal key ring file (referenced by either the **SSLClientKeyRing** or **SSLServerKeyRing** property) is searched for CA certificates instead.

Exception	Cause	Action
CertificateException	The certificate is not trusted because it is signed by a CA that is not in the CA key ring.	Check that all of the necessary CA certificates are present in the CA key ring file. Use the IBM Key Management tool supplied with MQIPT to add any missing CA certificates, taking care to obtain a copy of each CA certificate from a trustworthy source.

Exception	Cause	Action
CertificateExpiredException	1. The certificate has expired: its notAfter date has passed. 2. The system clock is set incorrectly.	1. Obtain a new certificate and insert it into the key ring file. If the certificate belongs to a Certificate Authority, place the new certificate into the CA key ring file. 2. Check that the UTC system clock is set to the correct time.
CertificateNotYetValidException	1. The certificate is being used prematurely: its notBefore date has not yet arrived. 2. The system clock is set incorrectly.	1. Check that the certificate has been generated and signed correctly. If your organization operates its own CA, the UTC system clock for the CA might be incorrect. 2. Check that the UTC system clock is set to the correct time.
CertificateParsingException	1. The certificate contains invalid DER data. 2. The certificate uses unsupported DER features.	Ensure the certificate has been correctly generated and can be viewed in the IBM Key Management tool supplied with MQIPT. Consider obtaining a new certificate with fewer certificate extensions.
CertificateRevokedException	Certificate revocation checking is enabled and the certificate was found to be revoked.	The certificate in question should not be trusted. Obtain a replacement certificate and ensure the new certificate and its private key are present in the key ring file.
CertPathBuilderException	The certificate chain was not signed by a recognized Certificate Authority.	1. If you are using CA-signed certificates, check that all root CA and intermediate CA certificates are present in the CA key ring file. 2. If you are using self-signed certificates, ensure that you have extracted a copy of the public part of the remote certificate and added it to the CA key ring file. Avoid using self-signed certificates in production environments.

Exception	Cause	Action
CertStoreException KeyStoreException	An error occurred reading a certificate from a key ring for one of the following reasons: 1. The key ring file is damaged. 2. The key ring file is missing. 3. The stored password does not match the key ring file password. 4. If the route is configured to use cryptographic hardware, MQIPT could not connect to the cryptographic hardware.	1. Ensure that the key ring file can be read and that all certificates can be viewed with the IBM Key Management tool. 2. Check that all key ring route properties refer to the correct file name. 3. Check that the stored key ring file password is correct. Use the mqiptPW tool to store the correct password. 4. If the route is configured to use cryptographic hardware, check the following: • The Java security properties file specifies that the IBMPKCS11Impl security provider is installed. • The Java security properties file contains the fully-qualified name of the configuration file that is used to initialize the IBMPKCS11Impl security provider. • The configuration file that is used to initialize the IBMPKCS11Impl security provider is valid.
SSLException: No available certificate or key corresponds to the SSL cipher suites which are enabled.	You must have a personal certificate with the correct type of key for the CipherSuites you are using. For example, CipherSuites whose names begin with SSL_ECDH_ECDSA_ require a certificate with an Elliptic Curve public key. The most commonly used CipherSuites require a certificate with an RSA public key.	Open the key ring file with the IBM Key Management tool. Under the Personal Certificates view, select each certificate in turn and view it. Click View Details and navigate to the Subject Public Key section to see the public key type. Then check the MQIPT SSLClientCipherSuites and SSLServerCipherSuites route properties to ensure that the appropriate CipherSuites are enabled.

Exception	Cause	Action
SSLException: No cipher suites in common SSLHandshakeException: No cipher suites in common	The handshake has failed to agree a CipherSuite because there is no compatible overlap between the sets of enabled CipherSuites at both ends of the connection. In particular, an outbound IBM MQ connection only enables a single cipher so SSLServer MQIPT routes are particularly likely to experience this error. This error can also occur when all three of the following conditions are true: - no CipherSuite is specified on the route - no suitable site certificate can be found in the key ring configured for the route - anonymous CipherSuites are disabled	Check the list of enabled CipherSuites in the MQIPT SSLClientCipherSuites and SSLServerCipherSuites route properties. Consider enabling additional CipherSuites. Consult the table provided to determine the correct CipherSuites to enable for each IBM MQ channel CipherSpec value. If no CipherSuite is specified on the route, check that key ring route properties refer to the correct key ring file, and that the key ring contains a personal certificate that MQIPT can use. If the route is configured to use cryptographic hardware, check that the tokenlabel attribute in the configuration file that is used to initialize the IBMPKCS11Impl security provider specifies the correct cryptographic device token label.

Managing MQIPT keystores

Use the **mqiptKeytool** command to manage certificates in keystores that IBM MQ Internet Pass-Thru (MQIPT) uses.

From IBM MQ 9.4.0, this command replaces the **mqiptKeycmd** command that is used to manage certificates in earlier versions of MQIPT.

Required keystore format for MQIPT

MQIPT supports keystores that use the PKCS #12 file format. When you use the **mqiptKeytool** command to manage MQIPT keystore, specify the **-storetype pkcs12** parameter to indicate that the keystore uses the PKCS #12 format.

MQIPT can also access certificates that are stored in cryptographic hardware that supports the PKCS #11 interface. The interface can also be used to manage certificates on PKCS #11 hardware. For more information, see [“Using PKCS #11 cryptographic hardware in MQIPT” on page 1100](#).

Encrypting the keystore password for MQIPT

Encrypt the keystore password in a format that MQIPT can use to access the file. For more information, see [“Encrypting a key ring password in MQIPT” on page 1086](#).

The stash file facility that IBM MQ supports is not supported by MQIPT. Use the **mqiptPW** command to encrypt the keystore password instead of using a stash file.

Examples

The following examples show how the **mqiptKeytool** command is used to manage certificates in a MQIPT keystore.

- The following command creates a self-signed personal certificate for testing purposes:

```
mqiptKeytool -genkeypair -keystore key.p12 -storetype pkcs12 -storepass password
              -alias mqipt -dname "CN=Test Certificate,OU=Sales,O=Example,C=US"
              -keyalg RSA -keysize 2048 -sigalg SHA256withRSA
```

The command creates a digital certificate with a 2048-bit RSA public key and a digital signature that uses RSA with the SHA-256 hash algorithm. The certificate and its associated public and private keys are stored in a PKCS #12 format keystore named `key.p12`. The keystore file is created if it does not exist.

When you create a certificate, choose a public key encryption algorithm, key size, and digital signature algorithm that are appropriate for your organization's security needs. For more information, see [“Digital certificate considerations for MQIPT”](#) on page 1092.

This example uses a self-signed certificate that is suitable for test purposes. In a production environment, use a certificate that is signed by a certificate authority instead.

- The following command creates a certificate request for a CA-signed certificate to be used for production purposes:

```
mqiptKeytool -certreq -keystore key.p12 -storetype pkcs12 -storepass password
              -alias mqipt -file cert.req
```

The command creates a Certificate Signing Request (CSR) in the PKCS #10 format. The CSR can be sent to a certificate authority to request a CA-signed certificate. The public and private key pair with the alias `mqipt` must be created before issuing this command, by issuing the **mqiptKeytool -genkeypair** command in the previous example.

- The following command receives the CA-signed personal certificate file that is named `cert.crt` into the keystore:

```
mqiptKeytool -importcert -keystore key.p12 -storetype pkcs12 -storepass password -file
cert.crt
```

Import the CA certificate of the CA that signed the personal certificate into the keystore by issuing the following command:

```
mqiptKeytool -importcert -keystore key.p12 -storetype pkcs12 -storepass password
              -file ca.crt -alias rootCA
```

Digital certificate considerations for MQIPT

Points to consider include certificate key size, the selection of an appropriate digital signature algorithm for the certificate, and digital certificate and CipherSuite compatibility.

Certificate key size considerations for MQIPT

The public key size depends upon your organization's security policy and on the encryption algorithm used. In general, larger key sizes are more secure. The following table lists the minimum key sizes that you should use:

Algorithm	Minimum key size (bits)
Elliptic Curve	256
RSA	2048

Specify the key size of your certificate by using the **-keysize** parameter when you create a certificate with the **mqiptKeytool** command.

Selecting an appropriate certificate digital signature algorithm

To prevent forgery of digital certificates, it is important to use a strong digital signature algorithm. When you create or request a certificate, take care to select a good algorithm.

You should avoid using old digital signature algorithms based on MD5 or SHA-1 as these algorithms are not considered secure. If possible, use one of the newer SHA-2 based digital signature algorithms such as SHA-256 with RSA (SHA256WithRSA).

Specify the signature algorithm by using the **-sigalg** parameter when you create a certificate with the **mqiptykeytool** command.

Digital certificate and CipherSuite compatibility in MQIPT

Not all CipherSuites can be used with all digital certificates. There are various types of CipherSuite, grouped by their CipherSuite name prefix. Each type of CipherSuite imposes different restrictions on the type of digital certificate that can be used. These restrictions apply to all MQIPT SSL/TLS connections, but are particularly relevant to users of Elliptic Curve cryptography. During a secure socket handshake, MQIPT automatically selects a personal certificate to identify itself that is appropriate for the negotiated CipherSuite. In most cases MQIPT automatically interoperates with the remote peer. However, in certain scenarios you might need to use a specific MQIPT CipherSuite to interoperate with a remote IBM MQ system.

The **mqiptykeytool** command can be used to create certificates with DSA, RSA, and Elliptic Curve public keys. Consult your certificate authority for advice on creating other types of certificate.

The type of digital certificate to use depends upon the type of CipherSuite you are using:

- CipherSuites with names that begin `SSL_ECDH_ECDSA_` and `SSL_ECDHE_ECDSA_` require a digital certificate with an Elliptic Curve public key.
- CipherSuites with names that contain *anon* are anonymous; they do not require a digital certificate to identify the remote peer. Such CipherSuites can avoid the overheads of certificate lifecycle management in networks where an alternative means of authentication is used, but in general, avoid their use due to the lack of authentication.
- Other CipherSuites require a digital certificate with an RSA public key.

Certificate exit in MQIPT

The purpose of a certificate exit is to validate an SSL/TLS peer certificate that is received by MQIPT.

You can configure an MQIPT route to act as an SSL/TLS client when it makes a new connection and to act as an SSL/TLS server when it receives a connection request. During the SSL/TLS handshaking process, an SSL/TLS client receives a peer certificate from the server, and the certificate can be used to authenticate the server. An SSL/TLS server can also receive a peer certificate from the client, and the certificate can be used to authenticate the client.

The certificate exit is called when MQIPT receives a peer certificate, allowing you to perform further validation. Any exceptions that are caught by the exit are caught by MQIPT and the connection request terminated. It is, therefore, good practice for the exit to catch all exceptions and to pass back an appropriate return code to MQIPT.

A sample is provided to show a certificate exit can be implemented for more information see [Using a certificate exit to authenticate an SSL/TLS server](#).

Note: MQIPT runs in a single Java Virtual Machine so a user-defined certificate exit might jeopardize the normal operation of MQIPT in one of these ways:

- Affect system resources
- Generate bottlenecks
- Degrade performance

You should test the effects of your certificate exit extensively before implementing it in a production environment.

The com.ibm.mq.ipt.exit.CertificateExit class in MQIPT

The com.ibm.mq.ipt.exit.CertificateExit class is an abstract class that must be implemented by the class that is defined with the SSLExitName property.

The class contains default implementations for running the exit and some public methods that you can optionally override, according to your requirements. The complete list of supported methods is as follows:

Methods

public int init(IPTTrace)

The init method is called by MQIPT when the exit is loaded by MQIPT and can be implemented to perform any initialization of the exit; for example, loading of data that is used during the validation process. The default implementation does nothing.

public int refresh(IPTTrace)

The refresh method is implemented to perform a refresh of any data; for example, reloading of any data for disk that is used during the validation process. This method is called when the MQIPT administrator has issued a refresh command. The default implementation does nothing.

public void close(IPTTrace)

The close method is implemented to perform any housekeeping when the route is about to be stopped or MQIPT is closing down. The default implementation does nothing.

public CertificateExitResponse validate(IPTTrace)

The validate method is called to perform validation of the peer certificate. The return object can be used to pass information back to MQIPT; for example, a return code and some text that can be added to the connection log. The default implementation returns a CertificateExitResponse with CertificateExitResponse.OK.

Supported methods for obtaining properties:

public int getListenerPort()

retrieves the route listener port - as defined by the ListenerPort property

public String getDestination()

retrieves the destination address - as defined by the Destination property

public int getDestinationPort()

retrieves the destination listener port address - as defined by the DestinationPort property

public String getClientIPAddress()

retrieves the IP address of the client making the connection request

public int getClientPortAddress()

retrieves the port address used by the client making the connection request

public boolean isSSLClient()

used to determine if the exit is being called as an SSL/TLS client or SSL/TLS server. If this returns true, the exit is on the client side of the connection, validating the certificate obtained from the server. If this returns false, the exit is on the server side of the connection, validating the certificate sent by the client. It is valid for a route to act as both an SSL/TLS server and an SSL/TLS client, decrypting and re-encrypting traffic. In this situation, although there is a single exit class, some instances of the class will be called as clients and some as servers. You can use isSSLClient to determine the situation for a given instance.

public int getConnThreadID()

used to retrieve the ID of the worker thread that is handling the connection request, which can be useful for debugging.

public String getChannelName()

retrieves the IBM MQ channel name that is used in the connection request. This is available only when the incoming request is not using SSL/TLS and MQIPT is acting as an SSL/TLS client.

public String getQMName()

retrieves the name of the IBM MQ queue manager used in the connection request. This is available only when the client request is not using SSL/TLS and MQIPT is acting as an SSL/TLS client.

public boolean getTimedout()

used by the exit to determine if the timeout has expired.

public IPTCertificate getCertificate()

retrieves the SSL/TLS certificate that needs to be validated.

public String getExitData()

retrieves the exit data, as defined by the SSLExitData property.

public String getExitName()

retrieves the exit name, as defined by the SSLExitName property.

The com.ibm.mq.ipt.exit.CertificateExitResponse class in MQIPT

This class is used to pass information back to MQIPT after a certificate has been validated.

Constructors**public CertificateExitResponse(int rc, string message)**

This constructor can be used to pass back a return code, and some message text. Possible reason codes are

- ExitRc.OK
- ExitRc.VALIDATE_ERROR
- ExitRc.VALIDATE_REJECTED

public CertificateExitResponse(int rc)

This constructor can be used to pass back a return code, with no message text. Possible reason codes are

- ExitRc.OK
- ExitRc.VALIDATE_ERROR
- ExitRc.VALIDATE_REJECTED

public CertificateExitResponse()

This constructor can be used to pass back return code ExitRc.OK, with no message text.

Methods**public String getVersion()**

This method returns the version of this class.

public String toString

This method will return a string representation of the response, for example, "Reason code: 4, Message: Failed CRL check."

The com.ibm.mq.ipt.exit.IPTCertificate class in MQIPT

This class contains the SSL/TLS certificate to be validated.

Methods**public int getVersion()**

This method returns the version of this class.

public byte [] getDerEncoding()

This method returns the ASN.1/DER encoding of the X.509 certificate, or NULL if there is an error.

public byte [] getPemEncoding()

This method returns the PEM (BASE64) encoding of the X.509 certificate, or NULL if there is an error.

public String getLabel()

This method returns the certificate label, or NULL if there is an error.

public String getName()

This method returns the Distinguished Name of the certificate, or NULL if not available. For example:

```
CN=Test Queue Manager,OU=Sales,O=Example,L=London,C=GB
```

public String getIssuerName()

This method returns the issuer's Distinguished Name of the certificate, or NULL if not available. For example:

```
CN=Certificate Authority,OU=Security,O=Example,L=New York,C=US
```

public IPTCertificate getSigner()

This method returns the signer certificate, or NULL if not available. For a self-signed certificate it will return a reference to itself.

public String toString()

This method returns a string representation of the certificate.

The com.ibm.mq.ipt.exit.IPTTrace class in MQIPT

The MQIPT trace functions provide entry and exit calls, that can be used on entry to and exit from a method. There are also various data calls to trace useful information.

Methods

public void entry(String fid)

Where *fid* is used to identify where the call was made, for example the class and method name.

This method writes an entry to the trace output file with the appropriate level of indentation to record the point at which the flow of control enters a method. This call is optional, but if it is used, a matching call to "exit(String)" must also be used within the same method.

public void exit(String fid)

Where *fid* is used to identify where the call was made, for example the class and method name.

This method writes an exit to the trace output file with the appropriate level of indentation to record the point at which the flow of control leaves a method. This method is used only when a call to "entry(String)" has previously been used within the same method.

public void exit(String fid, int rc)

Where *fid* is used to identify where the call was made, for example the class and method name, and *rc* is the numeric return code from the method. This trace method should be used to record the exit from methods that return an integer.

This method writes an exit to the trace output file with the appropriate level of indentation to record the point at which the flow of control leaves a method and the numeric return code from that method. This method is used only when a call to "entry(String)" has previously been used within the same method.

public void exit(String fid, boolean rc)

Where *fid* is used to identify where the call was made, for example the class and method name, and *rc* is the Boolean return code from the method. This trace method should be used to record the exit from methods that return a Boolean.

This method writes an exit to the trace output file with the appropriate level of indentation to record the point at which the flow of control leaves a method and the Boolean return code from that method. This method is used only when a call to "entry(String)" has previously been used within the same method.

public void data(String *fid*, String data)

Where *fid* is used to identify where the call was made, for example the class and method name.

This method writes some string data to the trace output file.

public void data(String *fid*, int data)

Where *fid* is used to identify where the call was made, for example the class and method name.

This method writes some integer data to the trace output file.

public void data(String *fid*, byte[])

Where *fid* is used to identify where the call was made, for example the class and method name.

This method writes some binary data to the trace output file.

Sample trace

To help diagnose problems in an exit, you can use the same tracing facility as MQIPT, alternatively you can implement your own tracing functions. If you decide to use the MQIPT trace functions there are entry and exit calls, that can be used on entry to and exit from a method. There are also various data calls to trace useful information as shown in the following example.

```
/**
 * This method is called to initialize the exit (for example, for
 * loading validation information) and place itself in a ready
 * state to validate connection requests.
 */
public int init(IPTTrace t) {
    final String fid = "MyExit.init";

    // Trace entry into this method
    t.entry(fid);

    // Trace useful information
    t.data(fid, "Starting exit - MQIPT version " + getVersion());

    // Perform initialization and load any data
    t.data(fid, "Ready for work");

    // Trace exit from this method
    t.exit(fid);

    return ExitRc.OK;
}
```

This method produces trace in the format shown in the following example:

```
16:36:48.625    14    5000-1s    -----{ ConnectionThread.setCertificateExit()
16:36:48.625    14    5000-1s    Creating instance of certificate exit
16:36:48.625    14    5000-1s    Calling init() of certificate exit
16:36:48.625    14    5000-1s    -----{ MyExit.init()
16:36:48.625    14    5000-1s    Starting exit - MQIPT version 2.1.0.0
16:36:48.625    14    5000-1s    Ready for work
16:36:48.625    14    5000-1s    -----{ MyExit.init() rc=0
16:36:48.625    14    5000-1s    -----{ ConnectionThread.setCertificateExit() rc=0
```

Certificate exit return codes in MQIPT

The return codes that MQIPT recognizes when calling a certificate exit in a number of different situations

The following return codes are recognized by MQIPT when calling a certificate exit in the following situations:

Return code	Description	init	validate	refresh
ExitRc.OK	Request completed successfully.	yes	yes	yes
ExitRc.INIT_ERROR	Init request failed, route will be disabled.	yes		
ExitRc.REFRESH_ERROR	Refresh request failed, route will be disabled.			yes
ExitRc.VALIDATE_ERROR	Validation process failed, connection request rejected.		yes	
ExitRc.VALIDATE_REJECTED	Validation request rejected, connection request rejected.		yes	

LDAP and CRLs in MQIPT

MQIPT supports use of a Lightweight Directory Access Protocol (LDAP) server to perform Certificate Revocation List (CRL) authentication on a digital certificate.

LDAP support has been implemented in a similar way to that in IBM MQ, as the same LDAP server can be used for both IBM MQ and MQIPT.

During the SSL/TLS handshake, the communicating partners authenticate each other with digital certificates. Authentication can include a check that the certificate received can still be trusted. Certification Authorities (CAs) revoke certificates for various reasons, including the following:

- The owner has moved to a different organization.
- The private key is no longer secret.

CAs publish revoked personal certificates in a Certificate Revocation List (CRL). CA certificates that have been revoked are published in an Authority Revocation List (ARL). Note that subsequent references to CRLs also apply to ARLs.

For further information about the use of LDAP servers with IBM MQ and about the management of CRLs and ARLs, see [Working with Certificate Revocation Lists and Authority Revocation Lists](#).

MQIPT can support up to two LDAP servers on each route. The first LDAP server is treated as the main server with the second LDAP server kept as a backup. The second server is only used if the main server cannot be reached. The backup server should be a mirror image of the main server.

Access to information stored on an LDAP server can be protected with a user ID and password by using the LDAP user ID and password properties. LDAP server passwords can be encrypted in the MQIPT configuration from IBM MQ 9.1.5. For more information on encrypting passwords to be used by MQIPT, see [“Encrypting stored passwords in MQIPT”](#) on page 1108.

When MQIPT loads a PKCS #12 token from a key ring file, any CA certificates are checked for CRL validity. If the CA certificate has an attached CRL, it is checked to see if it has expired and, if so, a newer CRL is retrieved from the LDAP server. Any CRL retrieved is loaded into the current token and attached to its CA certificate.

If there are no entries that match the given CA when a query is sent to the main LDAP server, it is assumed there are no CRLs for that CA and the backup server is not used. However, if the main LDAP server cannot be reached or does not return within a given time frame, the backup server is used. Any errors from the backup server cause the client connection to be ended. This action can be overridden by setting the **LDAPIgnoreErrors** property to true.

Any CRLs retrieved by MQIPT are kept in a cache and shared by all connections on that route. If a cached CRL has expired, the CRL is removed from the cache and a new one is retrieved from the LDAP server. If a new CRL is not available the connection is still refused.

A CRL retrieved from the LDAP server is also checked for expiry and a warning message is displayed (MQCPW001). The expired CRL is still loaded into the system and any connection requests referencing that CRL are refused. You should replace the expired CRL in the LDAP server with a current one.

The **LDAPCacheTimeout** property can be used to control how often the CRL cache is cleared. The default value is 1 day. Setting this value to 0 means the cache entries are not cleared until the route is restarted.

An expired CRL can be stored in a key ring file or on an LDAP server. If a new CRL has not been issued, further connection requests are refused. You can ignore expired CRLs by enabling the **IgnoreExpiredCRLs** property.

Note: If you enable either the **LDAPIgnoreErrors** property or the **IgnoreExpiredCRLs** property, a revoked certificate might be used to make an SSL/TLS connection.

Multi-valued certificate Distinguished Name OU properties in MQIPT

You can match multiple organizational unit (OU) values in certificate Distinguished Names.

The following route properties now support the matching of multiple OU values:

- **SSLClientDN_OU**
- **SSLClientSiteDN_OU**
- **SSLServerDN_OU**
- **SSLServerSiteDN_OU**

To match multiple OU values, use a comma as a separator in the route property value. For example:

```
SSLClientDN_OU=Sales, Europe
```

This matches certificates with both OU=Sales and OU=Europe. The OU values are matched in the same sequence as multiple OU values in IBM MQ SSLPEER filters.

Do not specify the same route property more than once in the [route] section. The correct way to match multiple OU values is to specify the property once, as shown in the preceding example. If you enter the same attribute more than once in the same `mqipt.conf` section, the last value takes effect. For example, the following entries would result in only matching Europe because the second line overrides the first:

```
SSLClientDN_OU=Sales
SSLClientDN_OU=Europe
```

If you must match a literal comma inside an OU value, insert a backslash (\) as an escape character immediately before the comma. For example:

```
SSLClientDN_OU=Sales\, Europe
```

This matches a single value: OU=Sales, Europe. A backslash that is not immediately followed by a comma matches a literal backslash.

If you are upgrading from a previous release of MQIPT and rely on the ability to match commas in OU values, you must insert backslash escape characters into the OU route properties in order to preserve the previous behavior.

Enabling deprecated protocols and Cipher Suites in MQIPT

By default, secure sockets protocols and cipher suites that are considered insecure are disabled in the Java runtime environment (JRE) supplied with MQIPT. These deprecated protocols and cipher suites must be enabled before they can be used.

About this task

If you are aware of the potential hazards, but still need to use one of the protocols or cipher suites that are considered insecure in MQIPT, follow this procedure to enable the protocol or cipher suite that you need to use.

Note: Deprecated protocols and cipher suites cannot be used with the TLS command port.

Procedure

1. Edit the `java.security` file.

The `java.security` is found in the following directory:

- In versions earlier than IBM MQ 9.4.4, `mqipt_path/java/jre/lib/security`.
-  From IBM MQ 9.4.4, `mqipt_path/java/jre/conf/security`.

where `mqipt_path` is the location where MQIPT is installed.

2. Add support to the JRE for a protocol or algorithm by removing the corresponding entry from the list of disabled algorithms in the `jdk.tls.disabledAlgorithms` property.
 - To add support for a protocol, remove the protocol from the list of disabled algorithms. For example, to add support for TLS 1.0, remove `TLSv1` from the list.
 - To add support for a cipher suite, remove the corresponding algorithms from the list of disabled algorithms. For example, to add support for the `SSL_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA` cipher suite, remove `3DES_EDE_CBC` and `DESede` from the list.
3. To enable SSL 3.0 in versions earlier than IBM MQ 9.4.4, you also need to set the system property `com.ibm.jsse2.disableSSLv3=false`.

If you are starting MQIPT from the command line using the **mqipt** command, you can set the property by using the environment variable **MQIPT_JVM_OPTIONS**. For example:

```
set MQIPT_JVM_OPTIONS=-Dcom.ibm.jsse2.disableSSLv3=false
```

 **Windows** If MQIPT is installed as a Windows service, you can set the property by defining a string value in the Windows registry under the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MQInternetPassThru` key. The value should have the following attributes:

Name

`MqiptJvmOptions`

Value data

`-Dcom.ibm.jsse2.disableSSLv3=false`

4. To enable SSL 3.0, TLS 1.0, or TLS 1.1 on a MQIPT route, add the corresponding protocol to the **SSLServerProtocols** or **SSLClientProtocols** route property.
5. Restart MQIPT for the changes to the JRE properties to come into effect.

Using PKCS #11 cryptographic hardware in MQIPT

MQIPT can access digital certificates that are stored in cryptographic hardware that supports the PKCS #11 interface.

Before you begin

Before you configure MQIPT to use cryptographic hardware, ensure that the cryptographic card, the card driver, and any associated support software are installed and functioning properly.

Support for PKCS #11 cryptographic hardware in MQIPT is provided by the IBM Java PKCS11 Cryptographic Provider (IBMPKCS11Impl provider). For more information about the IBMPKCS11Impl provider, and the list of cryptographic cards that Java 8 supports, see [IBM PKCS11 Cryptographic Provider](#).

About this task

You can store the personal certificates and CA certificates that MQIPT uses in a cryptographic hardware keystore. However, as a PKCS #11 device normally does not have enough space available to store many signer certificates, you might want to use a separate file-based keystore for CA certificates.

Follow this procedure to configure MQIPT to use certificates in a cryptographic hardware keystore.

Note:

- The use of cryptographic hardware with MQIPT is an IBM MQ Advanced capability. To use this capability, the local queue manager that is connected with the MQIPT route is also required to have IBM MQ Advanced, IBM MQ Appliance, IBM MQ Advanced for z/OS VUE, or IBM MQ Advanced for z/OS entitlement.
- The **mqiptKeytool** command does not currently support PKCS #11 cryptographic hardware. This procedure uses the **ikeycmd** command that is supplied with the Java runtime environment (JRE) instead to manage certificates in the cryptographic hardware. Ensure that the **ikeycmd** command in the JRE in the MQIPT installation path is used to run these commands.

Procedure

1. Create the configuration file that is used when the IBMPKCS11Impl provider is initialized.

Download sample configuration files for each of the hardware cryptographic cards that are supported by the IBMPKCS11Impl provider, and configure a sample for your system. The samples can be downloaded from the following topic in the IBM Documentation for Java: [Configuration file](#).

The configuration file is a text file, and must contain at least the following attributes:

name

The name suffix of the provider instance.

library

The fully qualified name of the PKCS #11 library that is supplied with the cryptographic hardware.

tokenlabel

The PKCS #11 cryptographic device token label.

For example, the configuration file might contain the following entries:

```
name = ITPKCS11Provider
library = /usr/lib64/pkcs11/PKCS11_API.so
tokenlabel = icatoken
```

2. Edit the Java security properties file, `java.security`, located in the `java/jre/lib/security` subdirectory of the MQIPT installation directory.

- a) If it is not already present in the file, add the IBMPKCS11Impl security provider.

For example, by adding the following line:

```
security.provider.12=com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl
```

- b) Add the fully qualified name of the configuration file after the provider name.

For example, if the configuration file you created in step “1” on page 1101 is called `/opt/mqipt/pkcs11.cfg`, add this path to the same line as the security provider:

```
security.provider.12=com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl /opt/mqipt/
pkcs11.cfg
```

3. Request a personal certificate for the cryptographic hardware.

Enter the following command to create a certificate request with the **ikeycmd** command:

```
MQIPT_INSTALL_DIRECTORY/java/jre/bin/ikeycmd -certreq -create
-crypto module_name -tokenlabel hardware_token
-pw password -label label -size key_size
-sig_alg algorithm -dn distinguished_name -file filename
```

where:

MQIPT_INSTALL_DIRECTORY

Is the MQIPT installation directory.

-crypto module_name

Specifies the fully qualified name of the PKCS #11 library that is supplied with the cryptographic hardware.

-tokenlabel *token_label*

Specifies the PKCS #11 cryptographic device token label.

-pw *password*

Specifies the password to access the cryptographic hardware.

-label *label*

Specifies the certificate label.

-size *key_size*

Specifies the key size. The value can be 512, 1024, 2048, or 4096.

-sig_alg *algorithm*

Specifies the asymmetric signature algorithm that is used for the creation of the entry's key pair. The value can be MD2_WITH_RSA, MD2WithRSA, MD5_WITH_RSA, MD5WithRSA, SHA1WithDSA, SHA1WithECDSA, SHA1WithRSA, SHA2/ECDSA, SHA224WithECDSA, SHA256_WITH_RSA, SHA256WithECDSA, SHA256WithRSA, SHA2WithECDSA, SHA3/ECDSA, SHA384_WITH_RSA, SHA384WithECDSA, SHA384WithRSA, SHA3WithECDSA, SHA5/ECDSA, SHA512_WITH_RSA, SHA512WithECDSA, SHA512WithRSA, SHA5WithECDSA, SHA_WITH_DSA, SHA_WITH_RSA, or SHAWithDSA. The default value is SHA256WithRSA.

-dn *distinguished_name*

Specifies the X.500 distinguished name enclosed in double quotation marks.

-file *filename*

Specifies the file name for the certificate request.

4. After the CA sends you the personal certificate, add the CA certificate to either the cryptographic hardware keystore or a CA keystore file, if it is not already present.

- To add the CA certificate to a PKCS #12 CA keystore file, enter the following command:

```
mqiptKeytool -importcert -keystore filename -storetype pkcs12 -storepass password
             -alias label -file cert_filename
```

where:

-keystore *filename*

Specifies the name of the CA keystore file. The keystore is created if it does not already exist.

-storepass *password*

Specifies the CA keystore password.

-alias *label*

Specifies the label of the CA certificate.

-file *cert_filename*

Specifies the name of the file that contains the CA certificate.

- To add the CA certificate to the cryptographic hardware, enter the following command:

```
MQIPT_INSTALL_DIRECTORY/java/jre/bin/ikeycmd -cert -add
             -crypto module_name -tokenlabel hardware_token
             -pw password -label label -file cert_filename
```

where:

MQIPT_INSTALL_DIRECTORY

Is the MQIPT installation directory.

-crypto *module_name*

Specifies the fully qualified name of the PKCS #11 library that is supplied with the cryptographic hardware.

-tokenlabel *token_label*

Specifies the PKCS #11 cryptographic device token label.

-pw *password*

Specifies the password to access the cryptographic hardware.

-label *label*

Specifies the label of the CA certificate.

-file *cert_filename*

Specifies the name of the file that contains the CA certificate.

5. Receive the personal certificate, provided by the CA, into the cryptographic hardware keystore.

Enter the following command to add the certificate to the cryptographic hardware keystore:

```
MQIPT_INSTALL_DIRECTORY/java/jre/bin/ikeycmd -cert -receive  
-crypto module_name -tokenlabel hardware_token  
-pw password -file filename
```

where:

MQIPT_INSTALL_DIRECTORY

Is the MQIPT installation directory.

-crypto *module_name*

Specifies the fully qualified name of the PKCS #11 library that is supplied with the cryptographic hardware.

-tokenlabel *token_label*

Specifies the PKCS #11 cryptographic device token label.

-pw *password*

Specifies the password to access the cryptographic hardware.

-label *label*

Specifies the certificate label.

-file *cert_filename*

Specifies the name of the file that contains the certificate to add.

If the CA certificate is stored in a CA keystore file, rather than in the cryptographic hardware, the command issues a warning that the certificate chain cannot be validated as the CA keystore cannot be accessed when the personal certificate is added into the cryptographic hardware keystore.

6. Encrypt the password to access the cryptographic hardware by using the **mqiptPW** command.

Enter the following command:

```
mqiptPW -sf encryption_key_file
```

where *encryption_key_file* is the name of a file that contains the password encryption key for your MQIPT installation. You do not need to specify the **-sf** parameter if your MQIPT installation is using the default password encryption key. Type in the password to access the cryptographic hardware to encrypt when prompted.

For more information about encrypting keystore passwords, see [“Encrypting a key ring password in MQIPT”](#) on page 1086.

7. If you added the CA certificate to a keystore file in step “4” on page 1102, follow the instructions in step “6” on page 1103 to encrypt the password for the CA keystore.
8. Edit the `mqipt.conf` configuration file.
 - a) Confirm that you have the appropriate entitlement to use this IBM MQ Advanced feature by setting the **EnableAdvancedCapabilities** global property to true.
 - b) Enable the use of the cryptographic hardware keystore on the route by setting one or more of the **SSLServerKeyRingUseCryptoHardware**, **SSLServerCAKeyRingUseCryptoHardware**, **SSLServerKeyRingUseCryptoHardware**, or **SSLServerKeyRingUseCryptoHardware** properties to true.

For more information about the properties to enable the use of cryptographic hardware on a route, see [MQIPT route properties](#).

You can also use cryptographic hardware with the TLS command port by setting the **SSLCommandPortKeyRingUseCryptoHardware** property to true.

- c) If you are using a keystore file for CA certificates, specify the location of the CA keystore by setting one or more of the **SSLServerCAKeyRing** or **SSLServerCAKeyRing** properties.
If you configure the route to use cryptographic hardware for the site certificate, and do not specify a CA key ring file, the cryptographic hardware keystore is used as the CA keystore.
- d) Specify the encrypted password to access the cryptographic hardware and the CA keystore by using the **SSLServerKeyRingPW**, **SSLServerCAKeyRingPW**, **SSLClientKeyRingPW**, **SSLClientCAKeyRingPW**, or **SSLCommandPortKeyRingPW** property.
Set the value of the **SSL*KeyRingPW** properties to the encrypted password that the **mqiptPW** command created.
- e) If the cryptographic hardware contains more than one personal certificate, specify which certificate MQIPT sends to the SSL/TLS server or client for authentication.
- For a SSL/TLS client route, specify which certificate is selected by using one or more of the **SSLClientSite*** properties.
 - For a SSL/TLS server route, specify which certificate is selected by using one or more of the **SSLServerSite*** properties.
 - For the TLS command port, specify which certificate is selected by using the **SSLCommandPortSiteLabel** property to specify the certificate label.

For more information about selecting certificates from a keystore, see [“Selecting certificates from a key ring file in MQIPT” on page 1086](#). The properties to select a certificate from a keystore are described in [MQIPT route properties](#).

For example, to use a cryptographic hardware keystore for the site certificate on a TLS server route, and a keystore file to store the CA certificates for the same route, add the following properties to the route definition:

```
SSLServerKeyRingUseCryptoHardware=true
SSLServerKeyRingPW=<mqiptPW>1!g0RdM4wft5d1rCgNMDEGag==!dZxhgQD2A8Ea0yeqawQvPg==
SSLServerCAKeyRing=/opt/mqipt/ssl/ca.pfx
SSLServerCAKeyRingPW=<mqiptPW>1!3Vdripiu6kMwn0sWRCVgT5g==!LH1tGLEg30FvN8+02Re0YA==
SSLServerSiteLabel=mqiptsite
```

9. Restart MQIPT.

Security exits in MQIPT

Use a security exit to control access to a target destination, as defined by the **Destination** route property. The security exit is called at the point when MQIPT receives a connection request from a client, but before it makes the connection to the target destination.

Based on the initial connection properties, the security exit decides whether the connection is allowed to complete.

When a route is started, the security exit is called in order to initialize and to make itself ready to process a connection request. The initialization process should be used to load any user data and prepare this data for quick and easy access, thereby minimizing the time taken to process a connection request.

Each route can have its own security exit.

- The **SecurityExit** property is used to enable/disable the user-defined security exit.
- The **SecurityExitName** property is used to define the class name of the user-defined security exit.
- The **SecurityExitPath** property is used to define the directory name containing the class file. If this property is not set, then it is assumed the class file will be found in the exits subdirectory. The **SecurityExitPath** can also define the name of a JAR file containing the user-defined security exit.
- The **SecurityExitTimeout** property is used by MQIPT to determine how long it should wait for a response from the security exit when validating a connection request.

See [MQIPT route properties](#) for details of the security exit properties.

MQIPT uses the `SecurityExit` class to call a user-defined security exit. This class must be extended by the user-defined security exit and most of its methods overridden to provide the functionality required. A `SecurityExitResponse` object is used to pass back data to MQIPT and this data is used by MQIPT to decide if the connection request should be accepted or rejected. The `SecurityExitResponse` object can also contain a new destination and destination port address, used to override the route defined by the security exit properties.

Three sample security exits are provided to show you how a security exit can be implemented.

- `SampleSecurityExit` shows how to control access to an IBM MQ queue manager, based on the name of the IBM MQ channel. It allows only a connection with a channel name starting with the string "MQIPT." See [Using a security exit](#) for more information.
- `SampleRoutingExit` allows dynamic routing of client connection requests to a pool of defined IBM MQ servers, each server hosting a queue manager of the same name and same attributes. The sample includes a configuration file that contains a list of server names. See [Routing client connection requests to IBM MQ queue manager servers by using security exits](#) for more information.
- `SampleOneRouteExit` allows dynamic routing to an IBM MQ queue manager that is derived from the IBM MQ channel name used in the connection request. The sample includes a configuration file that contains a map of queue manager names to server names. See [Dynamically routing client connection requests](#) for more information.

Note: MQIPT runs in a single JVM so a user-defined security exit might jeopardize the normal operation of MQIPT in one of these ways:

- Affect system resources
- Generate bottlenecks
- Degrade performance

You should test the effects of your security exit extensively before implementing it in a production environment.

The `com.ibm.mq.ipt.exit.SecurityExit` class in MQIPT

This class and its public methods must be extended by the user-defined security exit to get access to some common data and allow some MQIPT initialization to take place.

Before each method is called by MQIPT, some properties will be made available for the method to use. Their values can be retrieved by using the appropriate get methods defined in this class.

Methods

public int init(IPTTrace)

The following properties are available:

- listener port
- destination
- destination port
- version

The `init` method is called by MQIPT when a route is started. On return from this method the security exit must be ready to validate a connection request. Valid return codes are `ExitRc.OK` or `ExitRc.INIT_ERROR`.

public int refresh(IPTTrace)

The following properties are available:

- listener port
- destination
- destination port

The `refresh` method is called by MQIPT when the MQIPT configuration is refreshed. This action is usually performed when a property has been changed in the configuration file. MQIPT reloads all the properties from the configuration file to determine which properties have been changed and whether a route needs to be restarted.

This method should perform a reload of any external data it uses; that is, data loaded by the **`init`** method. Valid return codes are `ExitRc.OK` or `ExitRc.REFRESH_ERROR`.

`public void close(IPTTrace)`

The following properties are available:

- listener port
- destination
- destination port

The `close` method is called by MQIPT when it is stopping. This method should free any system resources that the exit has acquired during its operation. MQIPT waits for this method to complete before shutting down.

This method is also called if a security exit was previously enabled, but has now been disabled in the configuration file.

`public SecurityExitResponse validate(IPTTrace)`

The following properties are available:

- listener port
- destination
- destination port
- timeout
- client IP address
- client port address
- channel name
- queue manager name

The `validate` method is called by MQIPT when it receives a connection request to validate. The channel name and queue manager name will not be available if the **`SSLProxyMode`** property has been enabled, as this feature is only used to tunnel TLS data and therefore the data usually obtained from the initial data flow is unreadable.

The security exit must return a `SecurityExitResponse` object, containing the following information:

- reason code (must be set)
- new destination address (optional)
- new destination listener port address (optional)
- message (optional)

The reason code determines if the connection is accepted or rejected by MQIPT. The `newDestination` and `newDestinationPort` fields can optionally be set to define a new target queue manager. If you do not set these properties, the route **`Destination`** and **`DestinationPort`** properties defined in the configuration file are used. Any message returned is appended to the connection log file entry.

The following methods are supported for obtaining the values of MQIPT configuration properties:

`public int getListenerPort()`

retrieves the route listener port - as defined by the **`ListenerPort`** property

`public String getDestination()`

retrieves the destination address - as defined by the **`Destination`** property

`public int getDestinationPort()`

retrieves the destination listener port address - as defined by the **`DestinationPort`** property

public String getClientIPAddress()

retrieves the IP address of the client making the connection request

public int getClientPortAddress()

retrieves the port address used by the client making the connection request

public int getTimeout()

retrieves the timeout value. MQIPT will wait for the security exit to validate a request - as defined by the **SecurityExitTimeout** property

public int getConnThreadID()

retrieves the connection thread ID handling the connection request, which is useful for debugging purposes

public String getChannelName()

retrieves the IBM MQ channel name used in the connection request

public String getQMName()

retrieves the IBM MQ queue manager name used in the connection request

public boolean getTimedout()

can be used by the security exit to determine if the timeout has expired

The com.ibm.mq.ipt.exit.SecurityExitResponse class in MQIPT

This class is used to pass a response back to MQIPT from a user-defined security exit and is used to determine if the connection request should be accepted or rejected.

Objects of this type are only created in the validate method (see [“The com.ibm.mq.ipt.exit.SecurityExit class in MQIPT” on page 1105](#)). There are convenience constructors for creating these objects and there are methods for each property. See the sample security exits for more information.

Creating a default SecurityExitResponse object rejects the connection request.

Constructors**• public SecurityExitResponse (String dest, int destPort, int rc, String msg)**

where:

- dest is the new target destination
- destPort is the new destination port address
- rc is the reason code
- msg is a message that will be added to the connection log entry

• public SecurityExitResponse (String dest, int destPort, int rc)**• public SecurityExitResponse (int rc, String msg)****• public SecurityExitResponse (int rc)****Methods****public void setDestination(String dest)**

sets a new destination address for the connection request

public void setDestinationPort(int port) throws IPTException

sets a new destination listener port address for the connection request - throw an IPTException for an invalid port address

public void setMessage(String msg)

adds a message to the connection log record

public void setReasonCode(int rc)

sets the reason code for the connection request.

Security exit return codes in MQIPT

The return codes that MQIPT recognizes when calling a security exit in a number of different situations.

The following return codes are recognized by MQIPT when calling a security exit in the following situations:

Return code	Description	init	validate	refresh
ExitRc.OK	Request completed successfully.	yes	yes	yes
ExitRc.INIT_ERROR	Init request failed, route will be disabled.	yes		
ExitRc.REFRESH_ERROR	Refresh request failed.			yes
ExitRc.NOT_AUTHORIZED	Validation process failed, connection request rejected.		yes	
ExitRc.DISABLE_SSL	Validation request successful, connection to target will not use SSL or TLS.		yes	

Port number control in MQIPT

When using MQIPT, it is possible to restrict the range of local port number that are used when making an outgoing connection.

Set the **OutgoingPort** property on the route to specify the initial local port number, and set **MaxConnectionThreads** to specify the number of ports to be used. For example, if you set **OutgoingPort** to 1600 and **MaxConnectionThreads** to 20, then the range of local port numbers for that route is 1600 - 1619.

It is the responsibility of the MQIPT administrator to make sure that there are no conflicts of port numbers between routes.

If **OutgoingPort** is not defined, a default value of 0 means a system-allocated port number is used for each connection.

When using HTTP, the number of outgoing ports is twice as many as when not using HTTP. In the previous example, if the route used HTTP, the range of numbers would be 1600 - 1639.

See [Allocating port numbers](#) for more information.

Multihomed systems

When using a multihomed system, you can specify which IP address an outgoing connection will bind to by using the **LocalAddress** property. Host names are not supported on this property.

Encrypting stored passwords in MQIPT

The MQIPT configuration might include passwords to access various resources, as well as the password to access MQIPT using the command port. All these passwords should be protected by being encrypted.

About this task

All stored passwords for use by MQIPT should be protected by encrypting the password with the **mqiptPW** command. The encrypted passwords are stored as property values in the `mqipt.conf` configuration file. MQIPT is able to distinguish between encrypted passwords, plain text passwords, and file names in property values. You should encrypt all passwords stored for use by MQIPT in this way as it is the most secure protection method.

To improve the protection of key ring passwords, re-encrypt any key ring passwords that have previously been encrypted, using the latest protection method.

Note: The **SSLCommandPortKeyRingPW** property in the `mqipt.conf` configuration file, and the **SSLClientCAKeyRingPW** property in the **mqiptAdmin** properties file, cannot refer to password files. The values of these properties must be set to the encrypted password string output by the **mqiptPW** command.

If a plain text or weakly protected password is present in the MQIPT configuration, a warning message is issued either when MQIPT starts or when a route starts.

Use this procedure to encrypt a password to be stored for use by MQIPT using the latest protection method.

Procedure

1. Optional: Create a file containing the password encryption key, if you do not already have one.

MQIPT uses an encryption key to encrypt passwords. You can specify your own encryption key in a file. The file must contain at least one character, and only one line of text.

The same password encryption key is used to encrypt and decrypt all stored passwords for an instance of MQIPT. Therefore, you need only a single password encryption key file for each MQIPT installation.

You can use a different password encryption key to encrypt passwords stored in the **mqiptAdmin** properties file than the encryption key used to encrypt passwords in the MQIPT configuration.

If you are planning to run MQIPT as a service that is automatically started, you must create the password encryption key file with the default name of `mqipt_cred.key`, and place it in the MQIPT home directory.

You do not have to specify a password encryption key, however it is more secure to do so. If you do not specify your own encryption key, the default encryption key is used.

Note: You must ensure that appropriate file permissions are set on the password encryption key file to prevent any unauthorized users from reading the encryption key. Only the user that runs the **mqiptPW** command and the user under which MQIPT runs need authority to read the password encryption key.

2. Encrypt the password using the **mqiptPW** command.

The syntax of the **mqiptPW** command is described in [mqiptPW \(encrypt stored password\)](#).

If you created a password encryption key file in step “1” on [page 1109](#), specify the file name using the **-sf** parameter to **mqiptPW**. For example, the following command can be issued to encrypt a password using the encryption key in the file specified by the **-sf** parameter:

```
mqiptPW -sf /opt/mqipt/mqipt_password.key
```

3. Enter the password to be encrypted when prompted.

The encrypted password will be output by **mqiptPW**.

4. Copy the encrypted password into the appropriate property in the `mqipt.conf` configuration file, or the **mqiptAdmin** properties file.

For example, the following line specifies an encrypted password for the MQIPT access password:

```
AccessPW=<mqiptPW>1!QL+2Jvj/tigKK1D7Nz8Oqw==!AMDBef0UrmPf5i10uqV5MA==
```

5. Start MQIPT. If you created a password encryption key file in step “1” on [page 1109](#) with a name other than the default name, specify the name of the encryption key file when starting MQIPT.

You can specify the name of the password encryption key file using the **-sf** parameter when starting MQIPT. For example, issue the following command to start MQIPT using the encryption key in the file specified by the **-sf** parameter:

```
mqipt /opt/mqipt -sf /opt/mqipt/mqipt_password.key
```

For information about other methods to specify the password encryption key file name when starting MQIPT, see [Specifying the password encryption key](#).

You can specify the name of the password encryption key file for the **mqiPTAdmin** command using the **PasswordProtectionKeyFile** property in the **mqiPTAdmin** properties file.

Other security considerations for MQIPT

MQIPT has several additional functions that help a designer build a secure solution.

- If there are many clients in an internal network all trying to make outgoing connections, they can all go through an MQIPT located inside the firewall. The firewall administrator then has to grant external access only to the MQIPT computer.
- MQIPT can connect only to queue managers for which it has been explicitly configured in its configuration file, unless MQIPT is acting as a SOCKS proxy or is using a security exit.
- MQIPT verifies that the messages it receives and transmits are valid, and conform to the IBM MQ protocol. This helps prevent MQIPT being used for security attacks outside the IBM MQ protocol. If MQIPT is acting as an SSL/TLS proxy, when all IBM MQ data and protocols have been encrypted, MQIPT can only guarantee the initial SSL/TLS handshake.
- MQIPT allows channel exits to run their own end-to-end security protocols.
- You can restrict the total number of incoming connections by setting the **MaxConnectionThreads** property. This helps protect a vulnerable internal queue manager from denial of service attacks.

Configuration file

You must protect the MQIPT configuration file, `mqiPT.conf`, from being read by unauthorized users because it might contain sensitive information, such as the **AccessPW** password that controls remote administrative access to MQIPT. Protect all passwords specified in the configuration file by following the procedure in “Encrypting stored passwords in MQIPT” on page 1108. Also, ensure that `mqiPT.conf` is protected against unauthorized modification. Set the operating system file permissions for `mqiPT.conf` such that only the user account that runs MQIPT can read or update the file.

Command port

The MQIPT command ports accept administrative commands issued over the network to a remote instance of MQIPT by the **mqiPTAdmin** command.

MQIPT can be configured with one command port that is unsecured and one command port that is secured with TLS. Connections to the unsecured command port are not encrypted.

Note: Data sent over the network to the unsecured command port, including the MQIPT access password, can be visible to other users on the network.

You must consider whether you need to enable a command port, and assess the risks of allowing remote administration of MQIPT, before enabling either the unsecured or TLS command port. The **mqiPTAdmin** command can administer local instances of MQIPT that are running under the same user as the **mqiPTAdmin** command without using a command port. Therefore you might not need to enable a command port to administer local instances of MQIPT.

If the unsecured or TLS command port is enabled, you must prevent unauthorized access to the command port. For example, you should consider these points when securing access to the command port:

- Use a firewall to restrict the set of computers that can connect to the MQIPT command port.
- Enable authentication on the command ports by using the **AccessPW** and **RemoteCommandAuthentication** properties. For more information about enabling command port authentication, see [Command port authentication](#).
- Consider disabling remote shutdown with the **RemoteShutdown** property.
- Consider using the **CommandPortListenerAddress** and **SSLCommandPortListenerAddress** properties to configure the command ports to listen on a specific network interface.

For more information on using the **mqiptAdmin** command to administer MQIPT, see [Administering MQIPT by using the command line](#).

Connection logs in MQIPT

MQIPT provides a connection log facility that contains lists of all successful and unsuccessful connection attempts.

An entry is written to the connection log for each connection received or made by an MQIPT route, and for each administrative command received by MQIPT. The connection log is controlled by using the **ConnectionLog** and **MaxLogFileSize** properties. See [MQIPT global properties](#) for more information.

Each time MQIPT is started, a new connection log is created. For identification, the filename includes the current time stamp, for example:

```
mqiptYYYYMMDDHHmmSS.log
```

where

YYYY is the year

MM is the month

DD is the day

HH is the hours

mm is the minutes

SS is the seconds

When a connection log reaches the maximum size as determined by the **MaxLogFileSize** property, a backup file, `mqipt001.log`, is created. A maximum of two backup files are maintained (`mqipt001.log` and `mqipt002.log`).

An entry in the connection log represents each part of a connection request. A connection request that is received by MQIPT and the resulting new connection that MQIPT makes to the destination address appears as two log entries, and subsequently two further entries when each connection is ended.

Here is the connection log for a successful connection request:

```
Wed May 15 13:13:51 BST 2013 conn accept 127.0.0.1(3842) 127.0.0.1(5000) OK 5000-0
Wed May 15 13:13:51 BST 2013 conn conn 127.0.0.1(3843) localhost(3500) OK 5000-0
Wed May 15 13:13:52 BST 2013 conn close 127.0.0.1(3842) 127.0.0.1(5000) OK 5000-0
Wed May 15 13:13:52 BST 2013 conn close 127.0.0.1(3843) localhost(3500) OK 5000-0
```

Here is a connection log for a failed connection request:

```
Wed May 15 14:56:40 BST 2013 conn accept 127.0.0.1(4138) 127.0.0.1(7000) OK 7000-0
Wed May 15 14:56:40 BST 2013 conn close 127.0.0.1(4138) 127.0.0.1(7000) ERROR 7000-0
Unrecognized SSL handshake request '54'
```

Connection log entries

Each connection log entry contains the following information:

- The time when the entry was created.
- The type of entry. The value can be one of the following values:

admin

Administrative command

conn

Route connection

- The event that occurred. The value can be one of the following values:

accept

Connection request received

close

Connection closed

conn

Connection request to route destination

dspipt

Display MQIPT command received

nodata

No data received from caller

ping

Ping request received

status

Display status command received

refr

Refresh command received

stop

Stop command received

- The source network address and port number. The value LOCAL is displayed for administrative commands issued locally without using the command port.
- The destination network address and port number. This is not displayed for administrative commands issued locally without using the command port.
- The completion code. The value can be either OK or ERROR.
- The MQIPT thread identifier.
- An optional error message.

Configuring IBM MQ Internet Pass-Thru using containers

You can run IBM MQ Internet Pass-Thru (MQIPT) in a container. The base image used by the container image must use a Linux operating system that is supported.

Procedure

- A sample MQIPT Docker image is available in the mq-container GitHub repository. To build and run the container, follow the instructions in [IBM MQ Internet Pass-Thru on Docker](#).

What to do next

You can view running containers using the **docker ps** command. To view the console output of MQIPT running in a Docker container, use the **docker logs \${CONTAINER_ID}** command.

Configuring streaming queues

The streaming queues feature allows you to have a duplicate copy of each message put to a queue, delivered to a second queue. Configuration of streaming queues is done on a queue by queue basis.

Local and model queues have two new attributes related to streaming queues:

STREAMQ

This is the name of the queue to which streamed messages should be delivered. You should set the **STREAMQ** attribute to the name of another queue.

There are restrictions on which queues can be configured to stream messages to other queues, and there are restrictions on which queues can be set as a destination for streamed messages. For information on message streaming restrictions see [Stream queue restrictions](#).

STRMQOS

This is the quality-of-service to use when delivering streamed messages.

You can set the **STRMQOS** attribute to one of two values:

BESTEF

Best effort, which is the default value.

The queue manager attempts to deliver a copy of each message to the queue specified in the **STREAMQ** attribute. If there is a problem delivering the streamed message, this does not affect delivery of the original message.

MUSTDUP

The queue manager attempts to deliver a copy of each message to the streaming queue.

If there is a problem delivering the streamed message, the original message is not delivered to its queue and the application receives MQCC_FAILED along with an appropriate reason code.

See the [ALTER queues](#), [DEFINE queues](#), and [DISPLAY QUEUE MQSC](#) commands, and the [Change, Copy, and Create Queue](#), [Inquire Queue](#), and [Inquire Queue \(Response\) PCF](#) commands for more details.

If more than one copy of each message is required, you can configure the **STREAMQ** attribute to refer to the name of an IBM MQ alias queue whose target refers to an IBM MQ topic. When a message is put to the original queue, a copy of the message is published to the named topic.

You must ensure you have API or administered subscriptions to the topic object as each subscription receives a copy of the message. The message delivered to subscribers follows the same rules as other publish/subscribe messages. For example, each message has a new message identifier and the context fields of the MQMD are different from those in the original message. For more information on the similarities and differences between original and streamed messages see [Streamed messages](#).

Examples

Best effort example

In the following example, a local queue ORDERS.QUEUE is altered to put streamed messages to a second queue ANALYTICS.QUEUE. The BESTEF quality of service is used to ensure that if there is a problem putting the streamed message to ANALYTICS.QUEUE, for example, if ANALYTICS.QUEUE is full, the original message can still be put to the ORDERS.QUEUE.

This sort of configuration can be used to perform analytics on the orders being received, by analyzing the streamed messages, while the original messages are put to the order queue and are processed. A benefit of the streaming queue feature is that you can leave the streamed messages on the ANALYTICS.QUEUE waiting to be processed, without affecting the actual orders being satisfied by the business.

```
DEFINE QLOCAL (ANALYTICS.QUEUE)
```

```
ALTER QLOCAL (ORDERS.QUEUE) STRMQOS (BESTEF) STREAMQ (ANALYTICS.QUEUE)
```

Note: In the example **STRMQOS** has been set to BESTEF, although you can leave this attribute out of the **ALTER** command, because BESTEF is the default quality of service.

Must duplicate example

In this example, a local queue PAYMENTS.QUEUE is altered to put streamed copies of every message to another local queue AUDIT.QUEUE. It is important that every message put to the payment queue is streamed to the audit queue, so the MUSTDUP quality of service is used.

If there is a problem delivering the streamed message to its queue, the original message is not delivered either, and the application receives a suitable completion and reason code. The application must retry the put in the same way that it would if there was only a single queue involved.

```
DEFINE QLOCAL (AUDIT.QUEUE)
```

```
ALTER QLOCAL (PAYMENTS.QUEUE) STRMQOS (MUSTDUP) STREAMQ (AUDIT.QUEUE)
```

Notes:

1. It is not necessary for the streaming queue to exist when altering the original queue. However, it is important to note that because the quality of service being used is MUSTDUP, attempts to put messages to the original queue fail until you have defined the streaming queue.
2. When using a queue alias with a target of a topic object, if there are no subscribers, then delivery of the streamed message is still considered successful, and the original message is delivered to its queue.
3. If a streamed message cannot be delivered to its queue, the queue manager does not attempt to deliver it to the dead letter queue. However, if a streamed message is sent to a remote queue, when it travels over a channel to another queue manager, it can be delivered to a dead letter queue following existing dead letter rules.

Configuring the streaming queue

No additional configuration needs to be performed on the streaming queue. It receives messages from any queues that name it as a streaming queue. However it might be sensible to consider the attribute values configured on the streaming queue.

For example, if the original queue has a maximum depth of 100,000 and the streaming queue only has a maximum depth of 5000, streamed messages might be lost if STRMQOS is set to BESTEF, or fail at put time if STRMQOS is set to MUSTDUP, with an MQRC_Q_FULL error, even though the original queue has plenty of space left on it.

Consider which attributes on the streaming queue might need to be changed to have appropriate values, based on how the original queue is configured.

Related concepts

[Streaming queues](#)

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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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 IBM 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 (<https://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: